(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 241 677 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
18.09.2002 Bulletin 2002/38

(51) Int Cl.[7]: **G11C 16/34**, G11C 16/10

(21) Application number: **02005757.6**

(22) Date of filing: **13.03.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **14.03.2001 JP 2001073170**

(71) Applicant: **NEC CORPORATION**
Tokyo (JP)

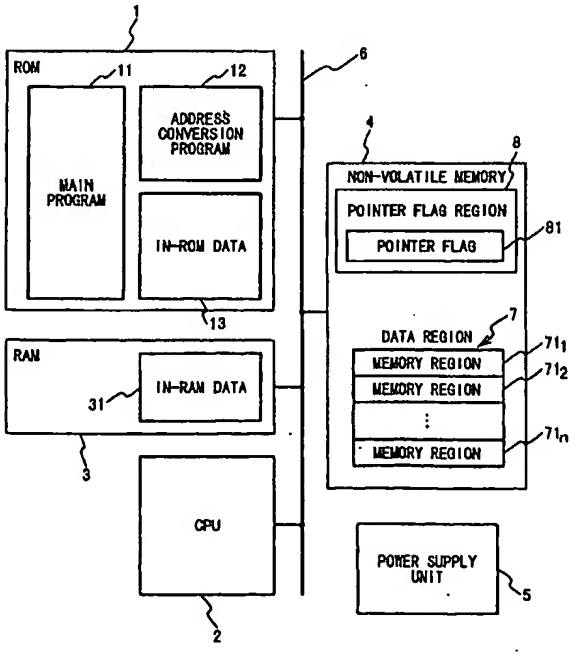
(72) Inventor: **Isogai, Hideo**
Tokyo (JP)

(74) Representative: **Glawe. Delfs. Moll**
Patentanwälte
Postfach 26 01 62
80058 München (DE)

(54) Semiconductor device using non-volatile memory in which data is written circularly

(57) A semiconductor device includes a non-volatile memory (4) and a control unit (1,2,3) which reserves n memory regions (n is an integer equal to or more than 2) for n of data as first to n-th memory regions in the non-volatile memory and writes the data in the first memory region, when the data is written in the non-volatile memory. The control unit sequentially and circularly writes the updated data as the latest data in the next region of said n memory regions, each time the data is updated.

Fig. 3



EP 1 241 677 A1

Printed by Jouve, 75001 PARIS (FR)

Description

Background of the Invention

1. Field of the Invention

[0001] The present invention relates to a semiconductor device. More particularly, the present invention relates to a semiconductor device having a built-in non-volatile memory in which a permissive number of times of rewrite is limited.

2. Description of the Related Art

[0002] Non-volatile memories are known such as EEPROM (Electrically Erasable Programmable Read Only Memory), a flash memory and FeRAM (Ferroelectric Random Access Memory). The non-volatile memory which can hold data without any power supply is suitable for a data memory for an IC card. In such a non-volatile memory, the permissive number of times of rewrite is presently limited. The permissive number of times of rewrite of the EEPRO is limited to about 10^5, the permissive number of times of rewrite of the flash memory is limited to about 10^4, and the permissive number of times of rewrite of the FeRAM is limited to about 10^{10}.

[0003] A data managing method is disclosed in Japanese Laid Open Patent Application (JP-A-Heisei 5-151097), in which circulation of a memory region to be accessed is carried out in accordance with the permissive number of times of rewrite for solving the above problem. Fig. 1 shows a conventional memory device to which the conventional data managing method is applied. The memory is limited in the permissive number of times of rewrite, and has a managing unit 101, a controller 102 and a plurality of memory regions 103. The managing unit 101 manages the number of times of rewrite of each memory region 103. The controller 102 selects one for data to be written from the memory regions 103 based on the permissive number of times of rewrite of each memory region 103. The controller 102 compares the numbers of times of rewrite of the memory regions 103, and selects the memory region with the minimum number of time of rewrite. The rewriting operation is carried out to the memory region with the minimum number of times of rewrite.

[0004] Figs. 2A to 2C show an operation of the conventional memory device in which the permissive number of times of rewrite is limited. It is supposed that a first semiconductor chip as a first memory region 103_1 of the memory regions 103 is already rewritten five times and a second semiconductor chip as a second memory region 103_2 of the memory regions 103 is not yet used. The number of times of rewrite of the first memory chip 103_1 and that of rewrite of the second memory chip 103_2 are compared with each other, and the second memory chip 103_2 with the smaller number of times is smaller is selected as a memory chip for the data to be rewritten. Contents of the first memory chip 103_1 are copied to the second memory chip 103_2. After that, the second memory chip 103_2 is used. The first memory chip 103_1 is collectively erased in units of chips. The first memory chip 103_1 is not used until the number of times of rewrite of the second memory chip 103_2 exceeds that of rewrite of the first memory chip 103_1.

[0005] In this way, the change of the memory chip 103 to be used is carried out so as to always use the memory chip with the minimum number of times of rewrite from among the plurality of memory chips 103. Thus, the memory chips 103 are accessed in the same number of times. If the number of the memory chips 103 is N, the lifetime of the memory chips 103 is elongated to N times.

[0006] In the conventional data managing method, it is necessary to refer to the numbers of times of rewrite of all of the memory regions 103 and further compare them in order to determine the memory region 103 to be used for the rewrite from among the plurality of memory regions 103. This results in the complex process to determine the memory region to be used for the rewrite.

[0007] In conjunction with the above description, a write method of a rewitable ROM is disclosed in Japanese Laid Open Patent application (JP-A-Heisei 9-265427). In this reference, the data region of the ROM is divided into blocks with a predetermined block length. Each of the blocks is allocated with an address. A CPU specifies one of the blocks based on a block address and data and writes a file in units of blocks. At this time, the CPU writes the file from an empty block when there is the empty block with a write start mark and from an optimal empty block when there is no empty block with the mark. After completion of the write of the file, the CPU searches an empty block in a downstream direction and writes the write start mark in the searched empty block.

[0008] Also, a semiconductor memory device is disclosed in Japanese Laid Open Patent application (JP-A-Heisei 8-96588). In this reference, the memory device is of a non-volatile type. A data storage section has at least one block, which can be independently written and erased, in a data storage region of the non-volatile memories. A block selecting section calculates a physical address from a logical address to specify one of the blocks. A search section searches an unused block from the data storage section. An erasure managing section manages initial erasure of the blocks. Data is written in the selected block and the block selecting section specifies the searched block as an object.

[0009] It is desirable to provide another technique for fictionally extending the lifetime of the non-volatile memory in which the permissive number of times of rewrite is limited. In particular, it is desirable to provide the technique for simply determining the memory region to be used for the rewrite and also performing the change of the used memory region, and thereby fictionally extending the lifetime of the non-volatile memory.

[0010] Also, in such a non-volatile memory, it is desir-

able that the data stored in the non-volatile memory is not damaged even if the power supply to the non-volatile memory is interrupted during the access to the non-volatile memory. If the non-volatile memory is used as a data memory for an IC card, the sudden interruption of the power supply is assumed to be frequently brought about. For example, if the non-volatile memory is used in a contact type IC card, the contact type IC card may be suddenly pulled out from a power supply terminal by a user. Moreover, if a non-contact type IC card to which power is supplied with electric wave is separated from a transmitter of the electric wave, there may be a case that a necessary power supply voltage cannot be kept. If the IC card is actually used in a dairy life, the interruption of the power supply is assumed to be brought about during the access to the non-volatile memory. If the non-volatile memory is used as the data memory for the IC card, it is important that the data stored in the non-volatile memory is not damaged even if the power supply is interrupted.

Summary of the Invention

[0011] Therefore, an object of the present invention is to provide the technique for fictionally extending the lifetime of the non-volatile memory in which the permissive number of times of rewrite is limited.

[0012] Another object of the present invention is to provide the technique for simply determining a memory region to be used for rewrite and performing the rotation on the memory region, and thereby fictionally extending the lifetime of the non-volatile memory in which the permissive number of times of rewrite is limited.

[0013] Still another object of the present invention is to provide the technique for reducing the risk that the data stored in the non-volatile memory is damaged if the power supply to the non-volatile memory is interrupted during the access to the non-volatile memory.

[0014] Yet still another object of the present invention is to provide the technique for detecting the interruption of the power supply to the non-volatile memory if the power supply to the non-volatile memory is interrupted.

[0015] In an aspect of the present invention, a semiconductor device includes a non-volatile memory and a control unit which reserves n memory regions (n is an integer equal to or more than 2) for n of data as first to n-th memory regions in the non-volatile memory and writes the data in the first memory region, when the data is written in the non-volatile memory. The control unit sequentially and circularly writes the updated data as the latest data in the n memory regions, and each time the data is updated.

[0016] The control unit may reserve a pointer region in the non-volatile memory to indicate an i-th one (i is an integer between 1 and n) of the first to n-th memory regions in which the latest data is written. In this case, the control unit may refer to the pointer region to read out the latest data from the i-th memory region.

[0017] Also, when the latest data should be further updated, the control unit may refer to the pointer region to read out the latest data from the i-th memory region, writes the updated data as the latest data in an (i+1)-th memory region if $1 \leqq i \leqq n-1$, and in the first memory region if $i = n$, and update the pointer region to indicate the (i+1)-th memory region or the first memory region.

[0018] Also, the pointer region may include (n-1) bit regions for the n memory regions. It is desirable that a (j-1)-th bit region (j is an integer between 2 and n) of the (n-1) bit regions indicates the j-th memory region in which the latest data has been written. In this case, the control unit may change a value of the (j-1)-th bit region from a first value to a second value when the latest data is written in the j-th memory region. Also, the control unit may reset values of the (n-1) bit regions to the first value when the latest data is written in the first memory region.

[0019] Also, the pointer region may include n bit regions corresponding to the n memory regions, and an i-th bit region of the n bit regions indicates the i-th memory region in which the latest data has been written. In this case, when the latest data is read out from an (i-1)-th memory region, updated and written in the i-th memory region, the control unit may change a value of the i-th bit region from a first value to a second value and changes a value of the (i-1)-th bit region from the second value to the first value.

[0020] Also, the control unit may reserve a power failure check region in the non-volatile memory. The power failure check region indicates whether power failure has occurred before the latest data is written in the i-th memory region. In this case, when the latest data written in the i-th memory region should be further updated, the control unit may refer to the pointer region and the power failure check region to determine an (i-1)-th memory region if the power failure check region indicates that the power failure has occurred when the latest data is written in the i-th memory region and the i-th memory region if the power failure check region indicates no power failure when the latest data is written in the i-th memory region. Also, the control unit may read out the latest data from the determined memory region, and writes the updated data as the latest data in the memory region next to the determined memory region.

[0021] Also, the control unit may determine n based on information supplied to the control unit.

[0022] Also, the above semiconductor device may be used for an IC card.

[0023] In another aspect of the present invention, a method of writing and updating data in a non-volatile memory, may be achieved by (a) reserving n memory regions (n is an integer equal to or more than 2) for n of data as first to n-th memory regions in a non-volatile memory, when the data is written in the non-volatile memory; by (b) reserving a pointer region in the non-volatile memory, the pointer region indicating that the latest data is written an i-th one (i is an integer between 1 and n) of the first to n-th memory regions; by (c) writing

the data in the first memory region as the latest data; by (d) sequentially and circularly writing the latest data in the n memory regions while the i increments by one each time the latest data is updated and the updated data is written as the latest data; and by (e) updating the pointer region to indicate that the latest data has been written in the i-th memory region, when the latest data has been written in the i-th memory region.

[0024] Also, the (d) sequentially and circularly writing step may be achieved by (f) referring to the pointer region to determine the i-th memory region, when the latest data is updated; by (g) reading out the latest data from the i-th memory region; and by (h) writing the updated data as the latest data in an (i+1)-th memory region. In this case, the (h) writing step may be achieved by writing the latest data in the (i+1)-th memory region if $1 \leqq i \leqq n-1$; and by writing the latest data in the first memory region if $i = n$.

[0025] The pointer region may include (n-1) bit regions for the n memory regions. In this case, the (e) updating step may be achieved by changing a value of the (j-1)-th bit region from a first value to a second value when the latest data is written in the j-th memory region. Also, the (e) updating step may further include resetting values of the (n-1) bit regions to the first value when the latest data is written in the first memory region.

[0026] Also, the pointer region may include n bit regions corresponding to the n memory regions. In this case, the (e) updating step may be achieved by changing a value of the i-th bit region from a first value to a second value and changes a value of the (i-1)-th bit region from the second value to the first value, when the latest data is read out from an (i-1)-th memory region, updated and written in the i-th memory region.

[0027] Also, the method may further include (i) reserving a power failure check region in the non-volatile memory. In this case, the (d) sequentially and circularly writing step further includes, when the latest data written in the i-th memory region should be further updated, referring to the pointer region and the power failure check region to determine an (i-1)-th memory region if the power failure check region indicates that the power failure has occurred when the latest data is written in the i-th memory region and the i-th memory region if the power failure check region indicates no power failure when the latest data is written in the i-th memory region; reading out the latest data from the determined memory region; and writing the updated data as the latest data in the memory region next to the determined memory region.

[0028] Also, the (a) reserving step may include determining the n based on given information.

[0029] Also, the above method may be achieved based on a program.

Brief Description of the Drawings

[0030]

Fig. 1 shows a conventional memory device;
Figs. 2A to 2C show an operation of the conventional memory device;
Fig. 3 shows a semiconductor device according to a first embodiment of the present invention;
Fig. 4 shows the configuration of a pointer flag and possible states of the pointer flag;
Figs. 5A to 5E show a process for updating data in the semiconductor device of the first embodiment;
Fig. 6 is a flowchart showing the process for updating the data in detail;
Figs. 7A to 7E show a process for reading out data from a data region;
Fig. 8 is a flowchart showing the process for reading out the data from the data region in detail;
Figs. 9A and 9B are diagrams showing the state of the data region when the power supply is interrupted during the update of the data region and the state of the pointer flag when the power supply is interrupted during the update of the pointer flag;
Fig. 10 shows a semiconductor device according to a second embodiment of the present invention;
Fig. 11 shows the configuration of a power interruption check pointer flag;
Figs. 12A to 12D show a process for updating data in the semiconductor device in the second embodiment;
Figs. 13A to 13D show a process for updating data in the semiconductor device of the second embodiment;
Fig. 14 shows a semiconductor device according to a third embodiment of the present invention; and
Figs. 15A to 15D shows a process for updating data in the semiconductor device of the third embodiment.

Description of the Preferred Embodiments

[0031] Hereinafter, a semiconductor device of the present invention will be described below with reference to the attached drawings.

(First Embodiment)

[0032] Fig. 3 shows a semiconductor device according to the first embodiment of the present invention. The semiconductor device is used for an IC card. The semiconductor device includes a ROM 1, a CPU 2, a RAM 3, a non-volatile memory 4 and a power supply unit 5. The ROM 1, the CPU 2, the RAM 3 and the non-volatile memory 4 are connected to each other through a bus 6.

[0033] Power is supplied from the power supply unit 5 through a power supply line (not shown) to the ROM 1, the CPU 2, the RAM 3 and the non-volatile memory

4. When the semiconductor device is used in a non-contact type IC card, a combination of a coil antenna and a regulator is selected as the power supply unit 5. When the semiconductor device is used in a contact type IC card, a power supply unit for converting a power supply voltage supplied from an external portion into a power supply voltage to be used in the semiconductor device is selected as the power supply unit 5.

[0034] The ROM 1 stores therein a main program 11, an address conversion program 12 and in-ROM data 13.

[0035] The main program 11 describes the whole operation of an IC card. The semiconductor device is operated in accordance with the main program 11.

[0036] The address conversion program 12 is a program for converting a virtual or logical address into a physical address. In the main program 11, the logical address is used to specify an address if an access to a memory space is carried out. The logical address is converted into the physical address using the address conversion program 12. The access is actually carried out to a region of the ROM 1, the RAM 2 and the non-volatile memory 4 which are specified by the physical address.

[0037] The in-ROM data 13 contains only data to be read out, of the data used in the main program 11.

[0038] The CPU 2 executes the main program 11 and the address conversion program 12, and then controls the whole operation of the semiconductor device of the present invention.

[0039] The RAM 3 stores therein in-RAM data 31. The in-RAM data 31 contains the data on which a re-written operation is frequently performed, of the data used in the main program 11.

[0040] The data which needs to be stored in the non-volatile manner and needs to be re-written are written to the non-volatile memory 4. The data are used in the main program 11. In the main program 11, it is supposed that data is stored in a logical address X. If the logical address X specifies a region of the non-volatile memory 4, and the region of the non-volatile memory 4 is assigned to store the data of the logical address X. In the non-volatile memory 4, the region assigned to store the data used in the main program 11 is hereafter referred to as the data region 7. If a plurality of data are used in the main program 11, a plurality of data region 7 are assigned in correspondence to them. However, a single data region 7 is shown in Fig. 3.

[0041] The data region 7 is provided with a plurality of memory regions 71_1 to 71_n. Here, n is an integer of 2 or more. The CPU 2 determines n when an instruction is inputted or information about the data such as a kind of data, and a location of the data in a data stream. However, the value of n may be fixed or predetermined. The CPU 2 reserves the data region 7 in the non-volatile memory 4 based on the determined value of n. When the data is written in the data region 7, any of the memory regions 71_{11} to 71_n is circularly selected. The data to be stored in the data region 7 is written in the memory region selected from among the memory regions 71_1 to 71_n. On the other hand, if the data is read out from the data region 7, the data is read out from the memory region, to which the latest writing operation is carried out, of the memory regions 71_1 to 71_n.

[0042] Another portion of the non-volatile memory 4 is assigned to a pointer flag region 8. A pointer flag 81 is stored in the pointer flag region 8. If a plurality of data regions 7 are provided in the non-volatile memory 4, one pointer flag 81 is provided for each data region 7. Fig. 3 shows only one pointer flag 81. The pointer flag 81 specifies one to be accessed of the memory regions 71_1 to 71_n. The pointer flag 81 is provided with (n-1) bits (81_1 to 81_{n-1}) as shown in Fig. 4. Here, n is the number of the memory regions 71_1 to 71_n. The number of bits 81_1 to 81_{n-1} of the pointer flag 81 is smaller by one than the number of the memory regions 71_1 to 71_n. Hereafter, a bit 81_k of the pointer flag 81 is represented as the lower order bit as the affix k is smaller.

[0043] The pointer flag 81 of the (n-1) bits is in either one of states 1 to n. The state 1 is the state in which all of the bits 81_1 to 81_{n-1} are 0. The state 2 is the state in which the bit 811 is "1" and the remaining bits 81_2 to 81_{n-1} are "0". Similarly, the state k is the state in which the bits 81_1 to 81_{k-1} are "1" and the remaining bits 81_k to 81_{n-1} are "0". Here, k is an integer between 1 and n. However, in a case of k=n, the state n is the state in which all of the bits 81_1 to 81_{n-1} are "1".

[0044] A memory region to be accessed of the memory regions 71_1 to 71_n is specified based on the state of the pointer flag 81. In case of the access to the data stored in the data region 7, the pointer flag 81 is referred by the CPU 2 using the address conversion program 12, and the memory region to be accessed of the memory regions 71_1 to 71_n is selected. Moreover, a physical address of the selected memory region is calculated by the address conversion program 12. Then, in the non-volatile memory 4, a portion specified by the physical address is accessed.

[0045] The correspondence between the state of the pointer flag 81 and the accessed memory region is as follows. If the pointer flag 81 is in the state 1, the data is read out from the memory region 71_1, and the data is written to the memory region 71_2. If the pointer flag 81 is in the state 2, the data is read out from the memory region 71_2, and the data is written to the memory region 71_3. Hereafter, similarly, if the pointer flag 81 is in the state k, the data is read out from the memory region 71_k, and the data is written to the memory region 71_{k+1}. However, if the pointer flag 81 is in the state n, the data is written to the memory region 71_1.

[0046] The number n of the memory regions 71_1 to 71_n is selected such that all of the (n-1) bits 81_1 to 81_{n-1} contained in the pointer flag 81 can be reset to "0" in response to one command outputted by the CPU 2. This provides the important role so that the data stored in the non-volatile memory 4 is not damaged if the power supply to the respective section of the semiconductor device from the power supply unit 5 is suddenly interrupt-

ed, as described later.

[0047] The operation of the semiconductor device in this embodiment will be described below.

[0048] At first, an operation for writing data to the data region 7 will be described. In the following description, it is supposed that the data region 7 is assigned to store the data of a logical address X in a memory space. Moreover, a data to be firstly stored as the data of the logical address X is supposed to be noted as a data 1.

[0049] As the data of the logical address X, the data 1 to be firstly written to the data region 7 is written to the memory region 71_1 without any condition. After the data 1 is written to the memory region 71_1, the pointer flag 81 is set in the state 1. That is, all of the bits 81_1 to 81_{n-1} included in the pointer flag 81 are reset to "0". Fig. 5A shows the states of the data region 7 and the pointer flag 81 after the completion of the process for writing the data 1.

[0050] After that, if the data of the logical address X is updated, the CPU 2 refers to the pointer flag 81 using the address conversion program 12, and determines which of the states 1 to n the pointer flag 81 is set to. If the pointer flag 81 is determined to be in the state k (k is an integer between 1 and n), the data to be stored in the data region 7 as the data of the logical address X is written to the memory region 71_{k+1} of the memory regions 71_1 to 71_n. After that, the state of the pointer flag 81 is shifted or changed from the state k to the state k+1. However, if the data of the logical address X is updated when the pointer flag 81 is in the state n, the data to be stored in the data region 7 as the data of the logical address X is written to the memory region 71_1, and the state of the pointer flag 81 is shifted from the state n to the state 1.

[0051] The above-mentioned processes will be actually described below with reference to Figs. 5A to 5E. As mentioned above, the data 1 to be firstly stored as the data of the logical address X is written to the memory region 71_1 as shown in Fig. 5A. When the data 1 is written to the memory region 71_1, the pointer flag 81 is set in the state 1.

[0052] If the data of the logical address X is updated from the data 1 to the data 2, the address conversion program 12 refers to the pointer flag 81. At this time, the pointer flag 81 is in the state 1. Thus, as shown in Fig. 5B, the memory region 71_2 is selected as the destination to which the data 2 is written. The data 2 is written to the selected memory region 71_2.

[0053] Subsequently, the bit 81_2 in the pointer flag 81 is rewritten from "0" to "1". That is, the pointer flag 81 is shifted from the state 1 to the state 2. Fig. 5B shows the states of the data region 7 and the pointer flag 81 after the pointer flag 81 is rewritten.

[0054] When the data of the logical address X is updated from the data 2 to the data 3, the data 3 is also similarly written to the memory region 71_3, as shown in Fig. 5C. After it is written, the pointer flag 81 is shifted to the state 3.

[0055] After that, when the data of the logical address X is sequentially updated from the data 3, to the data 4, the data 5, ..., and the data n, the data 4, the data 5, ..., and the data n after the update are written to the memory region 71_4, the memory region 71_5, ..., and the memory region 71_n, respectively. Similarly, the pointer flag 81 is sequentially shifted to the state 4, the state 5, ..., and the state n. After the completion of the process for writing the data n, the pointer flag 81 is in the state n.

[0056] Subsequently, if the data of the logical address X is updated from the data n to the data (n+1), the pointer flag 81 is in the state n as shown in Fig. 5D. Thus, as shown in Fig. 5E, the memory region 71_1 is selected as the destination to which the data (n+1) is written. The data (n+1) is written to the memory region 71_1, and the pointer flag 81 is shifted from the state n to the state 1.

[0057] In this way, each time the data of the logical address X is updated, one of the memory regions 71_1 to 71_n is circularly selected, and the data to be stored as the data of the logical address X is written to the selected memory region. In this way, the circulation selection of the written memory regions 71_1 to 71_n enables the permissive number of times of rewrite of the data of the logical address X to be apparently greater than the permissive number of times of rewrite which is physically allowed for the non-volatile memory 4.

[0058] Fig. 6 is a flowchart showing the writing operation to be performed on the data region 7 in detail, if the data of the logical address X is updated. The program for executing the operation shown in the flowchart of Fig. 6 is written in the address conversion program 12. The CPU 2 carries out the operation shown in the flowchart of Fig. 6, in accordance with the address conversion program 12.

[0059] At first, it is determined whether or not each of the bits 81_1 to 81_{n-1} is at "0". The determination is carried out in turn from the lower order bit 81_1 (Step S01).

[0060] Next, it is determined whether or not each of the bits 81_1 to 81_{n-1} includes the bit of "0" (Step S02).

[0061] If any of the bits 81_1 to 81_{n-1} is set to "0", a value i satisfying the following condition is determined (Step S03). The value i indicates a position of a least significant bit for which "0" is set. Condition: The bits 81_1 to 81_{n-1} of the bits 81_1 to 81_{i-1} are "1", and the remaining bits 81_i to 81_{n-1} are "0". However, if all of the bits 81_1 to 81_{n-1} are "0", it is determined to be i=1. The value i thus determined is any integer between 1 and n-1.

[0062] The fact that the bits 81z to 81_{i-1} are "1" and the remaining bits 81_i to 81_{n-1} are "0" implies that the pointer flag 81 is in the state i. If the pointer flag 81 is determined to be in the state i, the memory region 71_{i+1} of the memory regions 71_1 to 71_n is selected as the destination to which the data is written. The physical address of the selected memory region 71_{i+1} is calculated, and the data after the update of the logical address X is written to the region indicated by the physical address of the non-volatile memory 4 (Step S03).

[0063] After that, the bit 81_{i+1} among the bits 81_1 to

81_{n-1} is updated from "0" to "1" (Step S04). Thus, the pointer flag 81 is shifted from the state i to the state i+1. The process for updating the data of the logical address X is ended as mentioned above.

[0064] On the other hand, at the step S02, if all of the bits 81_1 to 81_{n-1} are at "1" and the bit of "0" is not found out, the head memory region 71_1 of the memory regions 71_1 to 71_n is selected as the destination to which the data is written. This implies that when the pointer flag 81 is in the state n, the memory region 71_1 is selected as the destination to which the data is written. The physical address of the selected memory region 71_1 is calculated, and the data after the update of the logical address X is written to the region indicated by the physical address of the non-volatile memory 4 (Step S05).

[0065] After that, all of the bits 81_1 to 81_{n-1} contained in the pointer flag 81 are reset to "0" (Step S06). The bits 81_1 to 81_{n-1} are collectively cleared in accordance with one command outputted from the CPU 2. If the address conversion program 12 instructs the CPU 2 to clear the bits 81_1 to 81_{n-1}, the CPU 2 outputs the command to instruct the non-volatile memory 4 to clear the bits 81_1 to 81_{n-1}. In accordance with the command, the non-volatile memory 4 clears all of the bits 81_1 to 81_{n-1} to "0". In this way, the bits 81_1 to 81_{n-1} are collectively cleared in accordance with the one command.

[0066] The operation for reading out the data from the data region 7 to which the data of the logical address X is written as mentioned above will be described below.

[0067] Figs. 7A to 7E are diagrams showing the operation for reading out the data from the data region 7. If the data of the logical address X is read out, the address conversion program 12 firstly refers to the pointer flag 81, and determines which of the states 1 to n the pointer flag 81 is set to.

[0068] If the pointer flag 81 is determined to be in the state j (j is an integer between 1 and n) of the states 1 to n, as the data of the logical address X, the data stored in the memory region 71_j among the memory regions 71_1 to 71_n is read out. As mentioned above, immediately after the data is written to the memory region 71_j, the pointer flag 81 is updated to the state j. Thus, the data written in the memory region 71_j is the latest data of the logical address X. Since the data is read out from the memory region 71_j when the pointer flag 81 is in the state j, the latest data of the logical address X is properly read out.

[0069] As shown in Fig. 7A, if all of the bits 81_1 to 81_{n-1} contained in the pointer flag 81 are "0", the pointer flag 81 is in the state 1. At this time, if the data of the logical address X is read out from the data region 7, the memory region 71_1 is selected as the source from which the data is read out. Then, the data of the logical address X is read out from the selected memory region 71_1.

[0070] Similarly, as shown in Fig. 7B, if only the bit 81_1 of the bits 81_1 to 81_{n-1} contained in the pointer flag 81 is "1" and the bits 81_2 to 81_{n-1} are "0", the pointer flag 81 is in the state 2. In this case, the memory region 71_2 is selected as the source from which the data is read out, and the data of the logical address X is read out from the memory region 71_2.

[0071] As shown in Figs. 7C to 7E, even if the pointer flag 81 is at another state, the data is similarly read out.

[0072] Fig. 8 is a flowchart showing the operation for reading out the data of the logical address X from the data region 7 in detail. The program for executing the operation shown in the flowchart of Fig. 8 is written in the address conversion program 12. The CPU 2 executes the operation shown in the flowchart of Fig. 8, in accordance with the address conversion program 12.

[0073] At first, it is determined whether or not each of the bits 81_1 to 81_{n-1} is "1" (Step S11). The determination is carried out in turn from the high order bit 81_{n-1}.

[0074] Next, it is determined whether or not each of the bits 81_1 to 81_{n-1} includes the bit of "1" (Step S12).

[0075] If any of the bits 81_1 to 81_{n-1} is set "1", j' satisfying the following condition is determined (Step S13). The value j' indicates the position of the most significant bit of the bits for which "1" is set.

Condition: The bits $81_{j'+1}$ to 81_{n-1} of the bits 81_1 to 81_{n-1} are "0", and the remaining bits 81_1 to $81_{j'}$ are "1". However, if all of the bits 81_1 to 81_{n-1} are "1", it is determined to be j'=n-1. The value j' thus determined is any integer between 1 and n-1.

[0076] Based on the determined j', the pointer flag 81 is determined to be in the state j (=j'-1). If the pointer flag 81 is determined to be in the state j, the memory region 71_j of the memory regions 71_1 to 71_n is selected as the source from which the data is read out. The physical address of the selected memory region 71_j is calculated, and the data of the logical address X is read out from the region indicated by the physical address of the non-volatile memory 4 (Step S13). The process for reading out the data of the logical address X is ended as mentioned above.

[0077] On the other hand, at the step S12, if all of the bits 81_1 to 81_{n-1} are "0" and the bit of "0" is not found out, the head memory region 71_1 of the memory regions 71_1 to 71_n is selected as the source from which the data is read out. This implies that when the pointer flag 81 is in the state 1, the memory region 71_1 is selected as the source from which the data is read out. The physical address of the selected memory region 71_1 is calculated, and the data of the logical address X is read out from the region indicated by the physical address of the non-volatile memory 4 (Step S14). The process for reading out the data of the logical address X is ended as mentioned above.

[0078] As mentioned above, in the semiconductor device in this embodiment, if data is stored in the non-volatile memory 4, a data region 7 is reserved for the data. The reserved data region 7 is comprised of the n memory regions 71_1 to 71_n (n is a natural number of 2 or more). If the data is updated and a data after the update is stored in the data region 7, one of the memory regions 71_1 to 71_n is circularly selected as the destination to

which the data is written. The data is written to the selected memory region.

[0079] At this time, the state of the pointer flag 81 is circularly shifted between the first to n-th states, in accordance with the selection of the memory regions 71_1 to 71_n. The respective bits 81_1 to 81_{n-1} are rewritten two times during one round of the selection of the memory regions 71_1 to 71_n. Thus, if the permissive number of times of rewrite of the non-volatile memory 4 is supposed to be N, the state of the pointer flag 81 can be changed by [n/2]×N times. Here, [x] is the maximum integer that does not exceed x.

[0080] As mentioned above, in the semiconductor device in this embodiment, the data stored in the data region 7 can be updated by [n/2]×N times. The permissive number of times of rewrite of the data stored in the data region 7 is greater than the permissive number of times of rewrite which is physically allowed for the non-volatile memory 4.

[0081] Moreover, in the semiconductor device in this embodiment, the risk that the data stored in the non-volatile memory 4 is damaged is small, even if the power supply to the respective section of the semiconductor device from the power supply unit 5 is suddenly interrupted during the access to the non-volatile memory 4.

[0082] At first, it is supposed that data is written to the data region 7 if the pointer flag 81 is at any state of the states 1 to n-1. Also, it is supposed that the pointer flag 81 is in the state i of the states 1 to n-1. The latest data at this time is stored in the memory region 71_i. If the data to be stored in the data region 7 is updated from that state and written to the data region 7, the data after the update is firstly written to the memory region 71_{i+1}. After that, the bit 81_i of the pointer flag 81 is updated from "0" to "1", and the pointer flag 81 is shifted from the state i to the state i+1.

[0083] At this time, as shown in Fig. 9A, it is supposed that the data is not normally written to the memory region 71_{i+1} because of the interruption of the power supply from the power supply unit 5 while the data is written to the memory region 71_{i+1}. Fig. 9A shows the data region 7 and the pointer flag 81 in case of i=2. In this case, the pointer flag 81 is kept at the original state i. Thus, when the data is read out from the data region 7 after that, the data is read out from the memory region 71_i. The latest data normally written is read out.

[0084] Moreover, as shown in Fig. 9B, it is supposed that the power supply from the power supply unit 5 is interrupted when the bit 81_i of the pointer flag 81 is updated from "0" to "1". In this case, whether the bit 81_i of the pointer flag 81 is set to "1" or "0" is determined as a probability.

[0085] If the bit 81_i of the pointer flag 81 is determined to be "1", this fact implies that the writing process is normally carried out as a result. Thus, there is no problem.

[0086] On the other hand, if the bit 81_i of the pointer flag 81 is determined to be "0", this fact implies that the writing process is not normally carried out. However, the pointer flag 81 is kept in the state i. Thus, when the data is read out from the data region 7 after that, the data is read out from the memory region 71_i. Then, the latest data is read out in which the writing process has been normally carried out.

[0087] In this way, even if the bit 81_i of the pointer flag 81 is determined to be "1" or "0", it is possible to read out the latest data in which the writing operation has been normally carried out.

[0088] Next, it is supposed that data is written to the data region 7 when the pointer flag 81 is in the state n. When the pointer flag 81 is in the state n, the memory region 71_1 is selected as the destination to which the data is written, and the data is written thereto. If the data is not normally written to the memory region 71_1 because of the interruption of the power supply from the power supply unit 5 while the data is written to the memory region 71_1, the pointer flag 81 is kept at the original state n. After that, when the data is read out from the data region 7, the data is read out from the memory region 71_n. Thus, even in this case, it is possible to read out the latest data in which the writing operation has been normally carried out.

[0089] After the data is written to the memory region 71_1, all of the bits 81_1 to 81_n in the pointer flag 81 are cleared to "0". Thus, the pointer flag 81 is shifted from the state n to the state 1. At this time, they are collectively cleared to "0", in accordance with to one command from the CPU 2. Hence, the risk that the content of the pointer flag 81 is damaged is extremely small when they are cleared to "0".

[0090] As mentioned above, even if the data is written to the data region 7 when the pointer flag 81 is at any one of the states 1 to n, the risk that the data stored in the non-volatile memory 4 is damaged is small in the semiconductor device in this embodiment.

(Second Embodiment)

[0091] Fig. 10 shows a semiconductor device according to the second embodiment of the present invention. The second embodiment differs from the semiconductor device in the first embodiment in that a power failure check pointer flag 82 is further provided in the pointer flag region 8, in addition to the pointer flag 81. The semiconductor device in the second embodiment uses the power failure check pointer flag 82. Thus, it is possible to detect the interruption of the power supply to the respective sections of the semiconductor device while the data is written to the data region 7.

[0092] Moreover, in the semiconductor device in the second embodiment, the contents of the main program 11 and the address conversion program 12 stored in the ROM 2 are modified. This modification corresponds to the fact that the power failure check pointer flag 82 is further installed in the pointer flag region 8.

[0093] The configurations of the other portions of the semiconductor device in the second embodiment are

the same as those of the semiconductor device in the first embodiment.

[0094] Fig. 11 shows the configuration of the power failure check pointer flag 82. The power failure check pointer flag 82 is provided with (n-1) interruption check bits 82_1 to 82_{n-1}. Here, n is the number of the memory regions 71_1 to 71_n of the data region 7, as mentioned above. As mentioned above, n is selected such that all of the (n-1) bits 81_1 to 81_{n-1} contained in the pointer flag 81 can be collectively cleared to "0" in accordance with the one command outputted by the CPU 2. Similarly, the (n-1) interruption check bits 82_1 to 82_{n-1} contained in the power failure check pointer flag 82 can be collectively cleared to "0" in accordance with the command outputted by the CPU 2.

[0095] The power failure check pointer flag 82 is also at any state of the states 1 to n, similarly to the pointer flag 81. The state 1 is the state in which all of the interruption check bits 82_1 to 82_{n-1} are "0". The state 2 is the state at which the interruption check bit 82_1 is "1" and the remaining interruption check bits 82_2 to 82_{n-1} are "0". Hereafter, similarly, the state k is the state at which the interruption check bits 82_1 to 82_{k-1} are "1" and the remaining interruption check bits 82_k to 82_{n-1} are "0". Here, k is an integer between 1 and n. However, in case of k=n, the state n is the state in which all of the interruption check bits 82_1 to 82_{k-1} are "1".

[0096] The pointer flag 81 and the power failure check pointer flag 82 are both kept at the same state in the usual condition. However, the interruption of the power supply during the writing process to the data region 7 causes a difference to be induced between the pointer flag 81 and the power failure check pointer flag 82. As described later, this difference enables the interruption of the power supply to be detected.

[0097] The operation of the semiconductor device in the second embodiment will be described below.

[0098] At first, the operation for writing a data to the data region 7 will be described. In the following description, it is supposed that the data region 7 is assigned to store the data of the logical address X in the memory space. Moreover, data to be firstly stored as the data of the logical address X is supposed to be noted as a data 1.

[0099] If the data 1 to be firstly stored as the data of the logical address X is written to the data region 7, the following writing process is carried out. At first, the power failure check pointer flag 82 is set in the state 1. That is, all of the interruption check bits 82_1 to 82_{n-1} are set to "0". Next, the data 1 is written to the data region 7. As the data of the logical address X, the data 1 to be firstly written to the data region 7 is written to the memory region 71_1 without any condition. After the data 1 is written to the memory region 71_1, the pointer flag 81 is set in the state 1. That is, all of the bits 81_1 to 81_{n-1} included in the pointer flag 81 are set to "0". Fig. 12A shows the states of the data region 7, the pointer flag 81 and the power failure check pointer flag 82 after the completion of the process for writing the data 1.

[0100] After that, if the data of the logical address X is updated, the following updating process is carried out. At first, the address conversion program 12 refers to the pointer flag 81 and the power failure check pointer flag 82, and determines which of the states 1 to n each of the pointer flag 81 and the power failure check pointer flag 82 is set to. The pointer flag 81 and the power failure check pointer flag 82 are usually in the states coincident with each other. If the pointer flag 81 and the power failure check pointer flag 82 are determined to be in the state k (k is an integer between 1 and n), the memory region 71_{k+1} of the memory regions 71_1 to 71_n is selected as a destination to which the data after the update is written.

[0101] Subsequently, the power failure check pointer flag 82 is shifted from the state k to the state k+1. In detail, the interruption check bit 82_k of the power failure check pointer flag 82 is shifted from "0" to "1". However, if the pointer flag 81 and the power failure check pointer flag 82 are in the state n, the power failure check pointer flag 82 is shifted from the state n to the state 1. That is, if the pointer flag 81 and the power failure check pointer flag 82 are in the state n, all of the interruption check bits 82_1 to 82_{n-1} are cleared to "0".

[0102] Next, the data after the update of the logical address X is written to the memory region 71_{k+1} of the memory regions 71_1 to 71_n. However, if the data of the logical address X is updated when the pointer flag 81 is in the state n, the data to be stored in the data region 7 as the data of the logical address X is written to the memory region 71_1.

[0103] After that, the pointer flag 81 is shifted from the state k to the state k+1. In detail, the bit 81_k of the pointer flag 81 is shifted from "0" to "1". However, when the pointer flag 81 is in the state n, the pointer flag 81 is shifted from the state n to the state 1. That is, when the pointer flag 81 is in the state n, $\$81_1$ to 81_{n-1} are all cleared to "0".

[0104] The process for updating the data of the logical address X is completed as mentioned above.

[0105] The above-mentioned processes will be actually described below. As mentioned above, the data 1 to be firstly stored as the data of the logical address X is written to the memory region 71_1 as shown in Fig. 12A. Both of the pointer flag 81 and the power failure check pointer flag 82 are set in the state 1.

[0106] If the data of the logical address X is updated from the data 1 to the data 2, the following processes are carried out. At first, the address conversion program 12 refers to the pointer flag 81 and the power failure check pointer flag 82. Both the pointer flag 81 and the power failure check pointer flag 82 are determined to be in the state 1. The memory region 71_2 of the memory regions 71_1 to 71_n is selected as the destination to which the data 2 is written. Then, as shown in Fig. 12B, the interruption check bit 82_1 of the power failure check pointer flag 82 is updated from "0" to "1". That is, the

power failure check pointer flag 82 is shifted from the state 1 to the state 2. Next, as shown in Fig. 12C, the data 2 is written to the memory region 71_2 selected as the destination to which the data is written. As shown in Fig. 12D, the bit 81_1 in the pointer flag 81 is updated from "0" to "1". That is, the pointer flag 81 is shifted from the state 1 to the state 2. The process for writing the data 2 is ended as mentioned above.

[0107] After that, when the data of the logical address X is sequentially updated to the data 3, ..., and the data n, the power failure check pointer flag 82 is sequentially shifted to the state 3, the state 4, ..., and the state n, similarly to the above-mentioned process. The data 3, the data 4, ..., and the data n after the update are written to the memory region 71_4, the memory region 71_5, ..., the memory region 71_n. Moreover, the pointer flag 81 is shifted sequentially to the state 3, the state 4, ..., the state n. After the completion of the process for writing the data n, the data n is written to the memory region 71_n, as shown in Fig. 13A, and the pointer flag 81 and the power failure check pointer flag 82 are in the state n.

[0108] Subsequently, if the data of the logical address X is updated from the data n to the data (n+1), the pointer flag 81 and the power failure check pointer flag 82 are firstly referred. The pointer flag 81 and the power failure check pointer flag 82 are determined to be in the state n. At this time, the memory region 71_1 of the memory regions 71_1 to 71_n is selected as the destination to which the data is written. As shown in Fig. 13B, all of the interruption check bits 82_1 to 82_{n-1} of the power failure check pointer flag 82 are cleared to "0", and the power failure check pointer flag 82 is shifted from the state n to the state 1. Next, as shown in Fig. 13C, the data (n+1) is written to the memory region 71_1 selected as the destination to which the data is written. As shown in Fig. 13D, all of the bits 81_1 to 81_{n-1} of the pointer flag 81 are cleared to "0". That is, the pointer flag 81 is shifted from the state n to the state 1. The process for writing the data (n+1) is ended as mentioned above.

[0109] The operation for reading out the data from the data region 7 to which the data of the logical address X is written as mentioned above will be described below.

[0110] If the data of the logical address X is read out, the address conversion program 12 firstly refers to the pointer flag 81, and determines which of the states 1 to n the pointer flag 81 is set to. At this time, the power failure check pointer flag 82 is not referred. As mentioned above, the power failure check pointer flag 82 is updated before the data is written to the memory regions 71_1 to 71_n. There is a case where the power failure check pointer flag 82 does not indicate the memory region of the memory regions 71_1 to 71_n in which latest data of the logical address X has been normally written, because the power supply is interrupted from the power supply unit 5 while the data is written to any of the memory regions 71_1 to 71_n. Thus, when the data of the logical address X is read out, the power failure check pointer flag 82 is not referred.

[0111] If the pointer flag 81 is determined to be in the state j of the states 1 to n (j is an integer between 1 and n), the data stored in the memory region 71_j of the memory regions 71_1 to 71_n is read out as the data of the logical address X. The process for reading out the data of the logical address X based on the state of the pointer flag 81 is the same as that of the first embodiment. Therefore, the detailed description is not carried out.

[0112] The programs for executing the above-mentioned operations are written in the main program 11 and the address conversion program 12 in this embodiment. The CPU 2 executes the above-mentioned operations in accordance with the main program 11 and the address conversion program 12.

[0113] In the semiconductor device in the second embodiment, if the data is stored in the non-volatile memory 4, the data region 7 is reserved for the data, similarly to the semiconductor device in the first embodiment. The reserved data region 7 is comprised of the n memory regions 71_1 to 71_n (n is a natural number of 2 or more). If the data is updated, one of the memory regions 71_1 to 71_n is circularly selected as the destination to which the data is written. The data is written to the selected memory region. Thus, the permissive number of times of rewrite of the data is fictionally greater than the physically permissive number of times of rewrite of the non-volatile memory 4.

[0114] Moreover, in the semiconductor device in this embodiment, similarly to the semiconductor device in the first embodiment, the risk that the data stored in the non-volatile memory 4 is damaged is small, even if the power supply to the respective sections of the semiconductor device from the power supply unit 5 is suddenly interrupted during the access to the non-volatile memory 4.

[0115] Moreover, in the semiconductor device in this embodiment, it is possible to detect the occurrence of the interruption of the power supply if the power supply from the power supply unit 5 is interrupted during the process for writing the data to the data region 7. If the power supply from the power supply unit 5 is interrupted during the update of the power failure check pointer flag 82, the write of the data to the memory regions 71_1 to 71_n and the update of the pointer flag 81, the difference is produced between the state of the pointer flag 81 and the state of the power failure check pointer flag 82. If the power supply from the power supply unit 5 is recovered, the CPU 2 determines whether or not the state of the pointer flag 81 is coincident with the state of the power failure check pointer flag 82, in accordance with the main program 11. Since the state of the pointer flag 81 is not coincident with the state of the power failure check pointer flag 82, the CPU 2 detects the interruption of the power supply from the power supply unit 5 during the process for writing the data to the data region 7. As mentioned above, in the semiconductor device in this embodiment, it is possible to detect the occurrence of the interruption of the power supply.

(Third Embodiment)

[0116] Fig. 14 shows a semiconductor device of the third embodiment according to the present invention. In the semiconductor device of the third embodiment, the pointer flag region 8 is not provided in the non-volatile memory 4. Instead, in the semiconductor device of the third embodiment, flags 72_1 to 72_n are added to the memory regions 71_1 to 71_n of the data region 7, respectively. Each of the flags 72_1 to 72_n stores one-bit data. The data region 7 to which the flags 72_1 to 72_n are added is hereafter referred as a data region 7'. Also, the whole of the flags 72_1 to 72_n is referred to as a flag set 72.

[0117] Moreover, in the semiconductor device of the third embodiment, the contents of the main program 11 and the address conversion program 12 stored in the ROM 2 are modified. This modification corresponds to the fact that the flags 72_1 to 72_n are added to the memory regions 71_1 to 71_n of the data region 7, respectively.

[0118] The flags 72_1 to 72_n indicates which of the memory regions 71_1 to 71_n of the data region 7' is accessed. Only one of the flags 72_1 to 72_n is set to "1", and the other flags are set to "0". The number of the combinations with regard to the values of the flags 72_1 to 72_n is n. That is, the number of possible states of the flag set 72 is n.

[0119] The correspondence relation between the values kept in the flags 72_1 to 72_n and the memory regions to be accessed is as follows. If the flag 72_1 of the flags 72_1 to 72_n is "1", the data is read from the memory region 71_1, and the data is written to the memory region 71_2. If the flag 72_2 of the flags 72_1 to 72_n is "1", the data is read out from the memory region 71_2, and the data is written to the memory region 71_3. Hereafter, similarly, if the flag 72_k of the flags 72_1 to 72_n is "1", the data is read out from the memory region 71_k, and the data is written to the memory region 71_{k+1}. However, if the flag 72_n of the flags 72_1 to 72_n is "1", the data is written to the memory region 71_1.

[0120] The operation of the semiconductor device of the third embodiment will be described below.

[0121] At first, the operation for writing data to the data region 7' will be described. In the following description, it is supposed that the data region 7' is assigned to store the data of the logical address X in the memory space. Moreover, it is supposed that the data to be firstly stored as the data of the logical address X' is data 1.

[0122] If the data 1 to be firstly stored as the data of the logical address X is written to the data region 7', the following writing process is carried out. At first, the data 1 is written to the memory region 71_1 without any condition. After the data 1 is written to the memory region 71_1, the flag 72_1 is set to the state "1", and the flags 72_2 to 72_n are remained set to "0". Fig. 15A shows the state of the data region 7' after the completion of the process for writing the data 1.

[0123] After that, when the data of the logical address X is updated, the following updating process is carried out. At first, the address conversion program 12 refers to the flags 72_1 to 72_n, and determines which of the flags 72_1 to 72_n is "1". If the flag 72_k of the flags 72_1 to 72_n is determined to be "1" (k is an integer between 1 and n), the memory region 71_{k+1} of the memory regions 71_1 to 71_n is selected as a destination to which the data after the update is written. However, if the flag 72_n is "1", the memory region 71_1 is selected as the destination to which the data after the update is written.

[0124] Next, the data of the logical address X after the update is written to the selected memory region 71_{k+1}. However, if the flag 72_n is at "1", the data of the logical address X after the update is written to the selected memory region 71_1.

[0125] After that, the flag 71_k is updated from "1" to "0", and the flag 71_{k+1} is updated from "0" to "1". However, if the flag 72_n is "1", the flag 71_1 is updated from "0" to "1".

[0126] The process for updating the data of the logical address X is ended as mentioned above.

[0127] The above-mentioned processes will be actually described below. As mentioned above, the data 1 to be firstly stored as the data of the logical address X is written to the memory region 71_1 as shown in Fig. 15A. The flag 72_1 is set to "1", and the flags 72_2 to 72_n are set to "0".

[0128] When the data of the logical address X is updated from the data 1 to the data 2, the following processes are carried out. At first, the flags 72_1 to 72_n are referred. If the fact that the flag 72_1 is to "1" and the flags 72_2 to 72_n are "0" is detected, the memory region 71_2 of the memory regions 71_1 to 71_n is selected as the destination to which the data is written. The data 2 is written to the memory region 71_2 selected as the destination to which the data is written. Next, the flag 72_1 is updated to "0", and the flag 72_2 is updated to "1". The process for updating the data of the logical address X from the data 1 to the data 2 is ended as mentioned above. Fig. 15B shows the states of the data region 7' and the flags 72_1 to 72_n when the updating process is ended.

[0129] After that, when the data of the logical address X is sequentially updated from the data 2 to the data 3, ..., and the data n, the data 3, the data 4, ..., and the data n after the update are written to the memory region 71_3, the memory region 71_4, ..., the memory region 71_n, respectively. Moreover, the flags of "1" of the flags 72_1 to 72_n are changed sequentially to the flag 72_3, the flag 72_4, ..., the flag 72_n. After the completion of the process for updating the data of the logical address X to the data n, the data n is written to the memory region 71_n, as shown in Fig. 15C, and the flag 72_n of the flags 72_1 to 72_n is set to "1", and the other flags 72_1 to 72_{n-1} are set to "0".

[0130] Subsequently, if the data of the logical address X is updated from the data n to the data (n+1), the flags 72_1 to 72_n are referred, and the memory region 71_1 of the memory regions 71_1 to 71_n is selected as the destination to which the data is written. The data (n+1) is writ-

ten to the memory region 71_1 selected as the destination to which the data is written. Next, the flag 72_1 is updated to "1", and the flag 72_n is updated to "0". The updating process for updating the data of the logical address X from the data n to the data (n+1) is ended as mentioned above. Fig. 15D shows the states of the data region 7' and the flags 72_1 to 72_n when the updating process is ended.

[0131] The operation for reading out the data from the data region 7' to which the data of the logical address X is written as mentioned above will be described below.

[0132] If the data of the logical address X is read out, the address conversion program 12 firstly refers to the flags 72_1 to 72_n, and determines which of the flags 72_1 to 72_n is "1". At this time, if the flag 72_k of the flags 72_1 to 72_n is determined to be "1", the data stored in the memory region 71_k of the memory regions 71_1 to 71_n is read out as the data of the logical address X. The operation for reading out the data from the data region 7' is ended as mentioned above.

[0133] The programs for executing the above-mentioned operations is written in the main program 11 and the address conversion program 12 in this embodiment. The CPU 2 executes the above-mentioned operations in accordance with the main program 11 and the address conversion program 12.

[0134] In the semiconductor device in the third embodiment, if the data is stored in the non-volatile memory 4, the data region 7 is reserved for the data, similarly to the semiconductor device in the first or second embodiments. The reserved data region 7 is comprised of the n memory regions 71_1 to 71_n (n is a natural number of 2 or more). When the data is updated, one of the memory regions 71_1 to 71_n is circularly selected as the destination to which the data is written. The data is written to the selected memory region. Thus, the permissive number of times of rewrite of the data is fictionally greater than the physically permissive number of times of rewrite of the non-volatile memory 4.

[0135] The present invention provides the technique for fictionally extending the lifetime of the non-volatile memory in which the permissive number of times of rewrite is limited.

[0136] Also, the present invention provides the technique for simply determining a memory region to be used for the rewrite and performing the rotation on the memory regions, and thereby fictionally extending the lifetime of the non-volatile memory in which the permissive number of times of rewrite is limited.

[0137] Also, the present invention provides the technique for reducing the risk that the data stored in the non-volatile memory is damaged if the power supply to the non-volatile memory is interrupted during the access to the non-volatile memory.

[0138] Moreover, the present invention provides the technique for detecting the occurrence f the interruption of the power supply to the non-volatile memory if the power supply to the non-volatile memory is interrupted.

Claims

1. A semiconductor device comprising:

   a non-volatile memory (4); and
   a control unit (1, 2, 3) which reserves n memory regions (n is an integer equal to or more than 2) (71_1 to 71_n) for n of data as first to n-th memory regions in said non-volatile memory and writes the data in said first memory region, when the data is written in said non-volatile memory, and each time the data is updated, sequentially and circularly writes the updated data as the latest data in said n memory regions.

2. The semiconductor device according to claim 1, wherein said control unit reserves a pointer region (81) in said non-volatile memory to indicate an i-th one (i is an integer between 1 and n) of said first to n-th memory regions in which the latest data is written.

3. The semiconductor device according to claim 2, wherein said control unit refers to said pointer region to read out the latest data from said i-th memory region.

4. The semiconductor device according to claim 2 or 3, wherein when the latest data should be further updated, said control unit refers to said pointer region to read out the latest data from said i-th memory region, writes the updated data as the latest data in an (i+1)-th memory region if $1 \leqq i \leqq n-1$, and in said first memory region if i = n, and updates said pointer region to indicate said (i+1)-th memory region or said first memory region.

5. The semiconductor device according to any of claims 2 to 4, wherein said pointer region comprises (n-1) bit regions (81_1 to 81_{n-1}) for said n memory regions, and an (j-1)-th bit region (j is an integer between 2 and n) of said (n-1) bit regions indicates said j-th memory region in which the latest data has been written.

6. The semiconductor device according to claim 5, wherein said control unit changes a value of said (j-1)-th bit region from a first value to a second value when the latest data is written in said j-th memory region.

7. The semiconductor device according to claim 6, wherein said control unit resets values of said (n-1) bit regions to the first value when the latest data is written in said first memory region.

8. The semiconductor device according to any of claims 2 to 4, wherein said pointer region comprises

n bit regions (72_1 to 72_n) corresponding to said n memory regions, and an i-th bit region of said n bit regions indicates said i-th memory region in which the latest data has been written.

9. The semiconductor device according to claim 8, wherein when the latest data is read out from an (i-1)-th memory region, updated and written in said i-th memory region, said control unit changes a value of said i-th bit region from a first value to a second value and changes a value of said (i-1)-th bit region from the second value to the first value.

10. The semiconductor device according to any of claims 2 to 8, wherein said control unit reserves a power failure check region (82) in said non-volatile memory, and

said power failure check region indicates whether power failure has occurred before the latest data is written in said i-th memory region.

11. The semiconductor device according to claim 10, wherein when the latest data written in said i-th memory region should be further updated, said control unit refers to said pointer region and said power failure check region to determine an (i-1)-th memory region if said power failure check region indicates that the power failure has occurred when the latest data is written in said i-th memory region and said i-th memory region if said power failure check region indicates no power failure when the latest data is written in said i-th memory region, reads out the latest data from said determined memory region, and writes the updated data as the latest data in said memory region next to said determined memory region.

12. The semiconductor device according to any of claims 1 to 11, wherein said control unit determines said n based on information supplied to said control unit.

13. An IC card using the semiconductor device according to any of claims 1 to 12.

14. A method of writing and updating data in a non-volatile memory, comprising the steps of:

(a) reserving n memory regions (n is an integer equal to or more than 2) for n of data as first to n-th memory regions in a non-volatile memory, when the data is written in said non-volatile memory;
(b) reserving a pointer region in said non-volatile memory, said pointer region indicating that the latest data is written an i-th one (i is an integer between 1 and n) of said first to n-th memory regions;
(c) writing the data in said first memory region as the latest data;
(d) sequentially and circularly writing the latest data in said n memory regions while said i increments by one each time the latest data is updated and the updated data is written as the latest data; and
(e) updating said pointer region to indicate that the latest data has been written in said i-th memory region, when the latest data has been written in said i-th memory region.

15. The method according to claim 14, wherein said (d) sequentially and circularly writing step comprises the step of:

(f) referring to said pointer region to determine said i-th memory region, when said latest data is updated;
(g) reading out the latest data from said i-th memory region; and
(h) writing the updated data as the latest data in an (i+1)-th memory region.

16. The method according to claim 15, wherein said (h) writing step comprises the steps of:

writing the latest data in said (i+1)-th memory region if $1 \leqq i \leqq n-1$; and
writing the latest data in said first memory region if $i = n$.

17. The method according to any of claims 14 to 16, wherein said pointer region comprises (n-1) bit regions for said n memory regions, and

said (e) updating step comprises the step of:
changing a value of said (j-1)-th bit region from a first value to a second value when the latest data is written in said j-th memory region.

18. The method according to claim 17, wherein said (e) updating step further comprises the step of:

resetting values of said (n-1) bit regions to the first value when the latest data is written in said first memory region.

19. The method according to any of claims 14 to 16, wherein said pointer region comprises n bit regions corresponding to said n memory regions, and

said (e) updating step comprises the step of:
when the latest data is read out from an (i-1)-th memory region, updated and written in said i-th memory region, changing a value of said i-th bit region from a first value to a second value

and changes a value of said (i-1)-th bit region from the second value to the first value.

20. The method according to any of claims 14 to 19, further comprising the step of:

(i) reserving a power failure check region in said non-volatile memory, and
wherein said (d) sequentially and circularly writing step further comprises the steps of:
when the latest data written in said i-th memory region should be further updated, referring to said pointer region and said power failure check region to determine an (i-1)-th memory region if said power failure check region indicates that the power failure has occurred when the latest data is written in said i-th memory region and said i-th memory region if said power failure check region indicates no power failure when the latest data is written in said i-th memory region;
reading out the latest data from said determined memory region; and
writing the updated data as the latest data in said memory region next to said determined memory region.

21. The method according to any of claims 14 to 20, wherein said (a) reserving step comprises the step of:

determining said n based on given information.

22. A program for executing said method according to any of claims 14 to 21.

Fig. 1 PRIOR ART

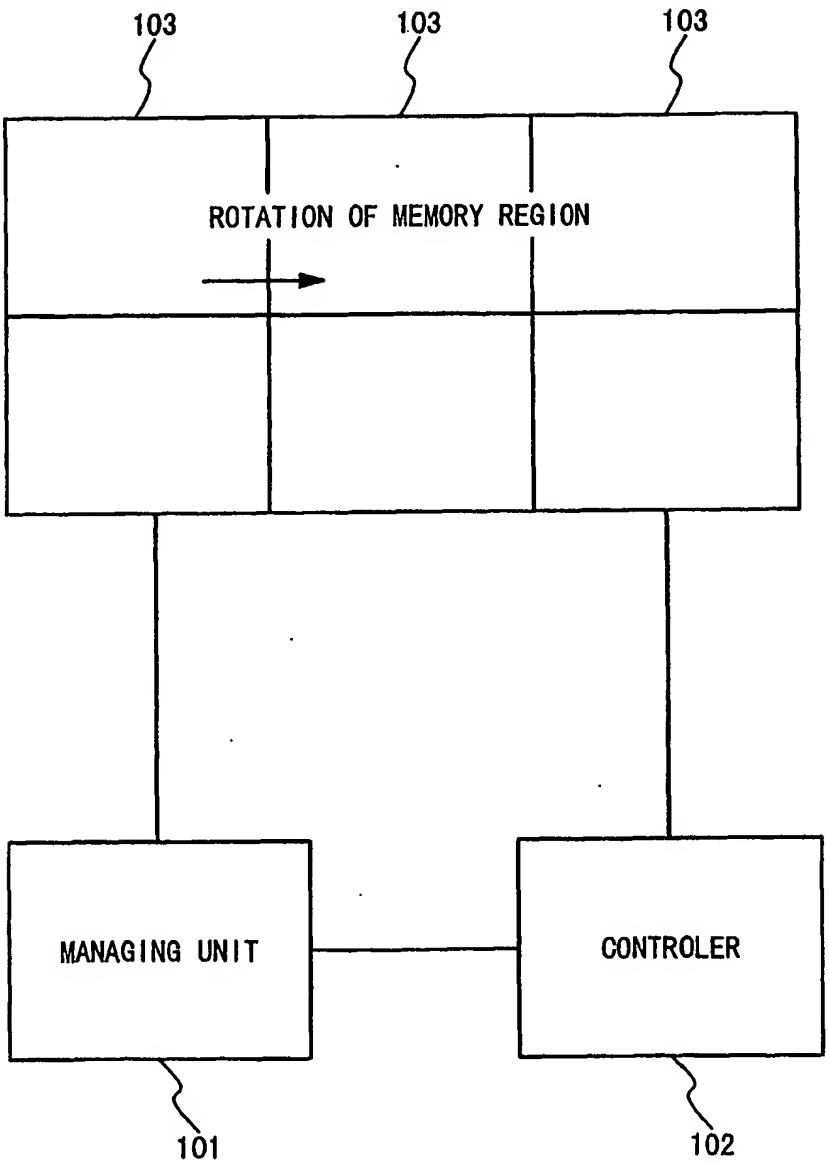


Fig. 2A
PRIOR ART

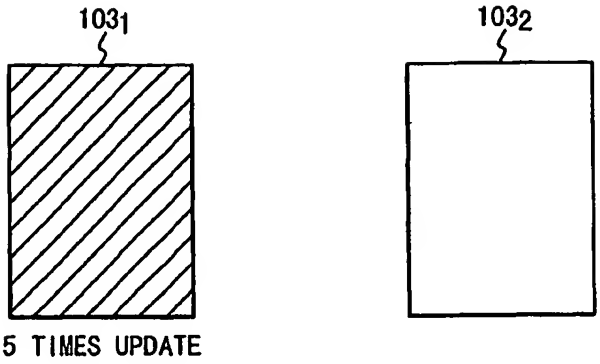


Fig. 2B
PRIOR ART

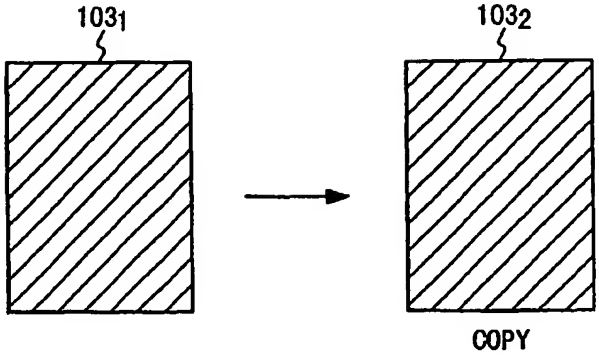


Fig. 2C
PRIOR ART

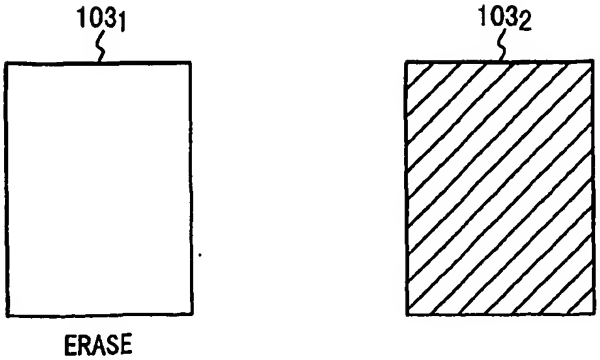


Fig. 3

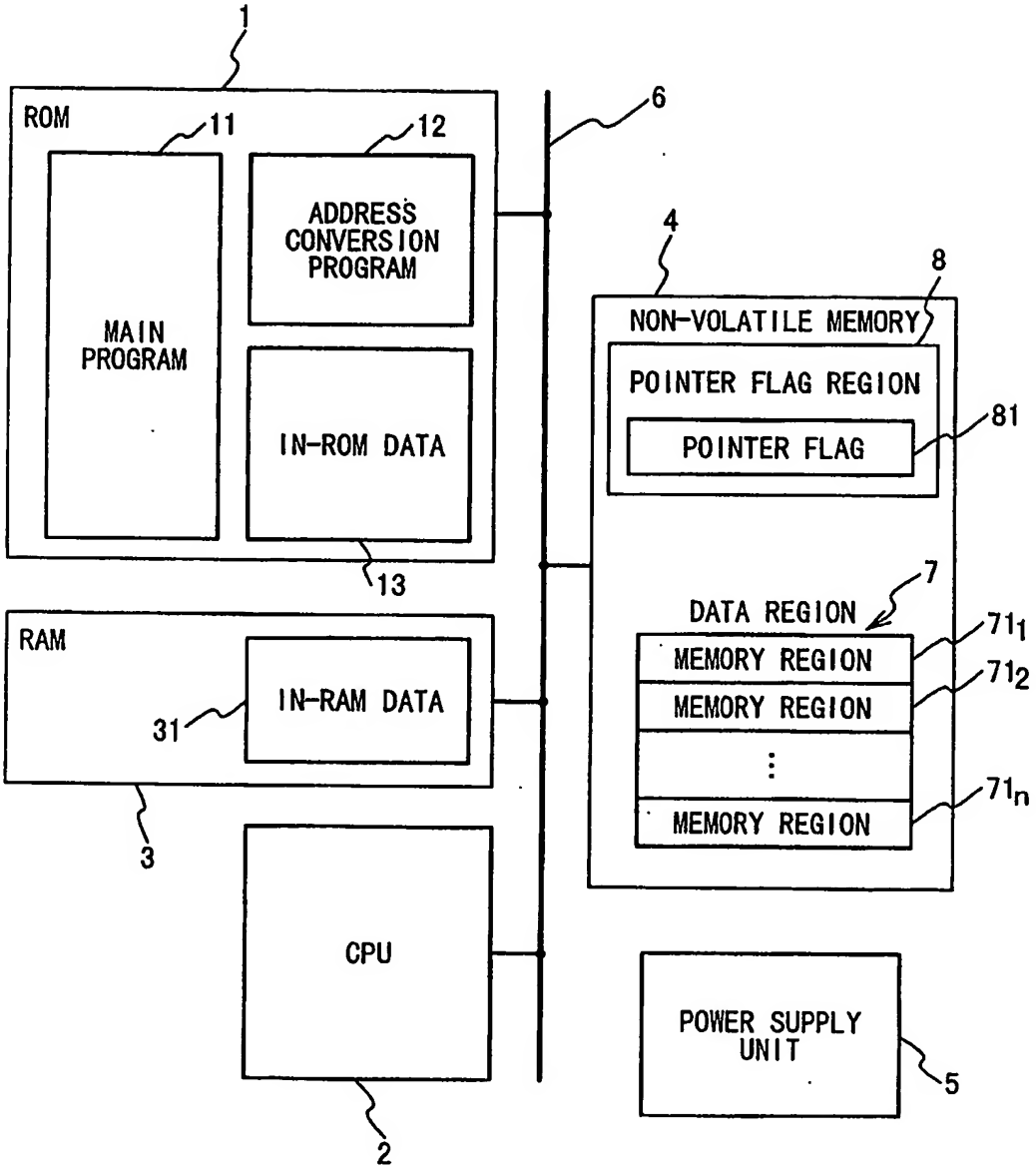
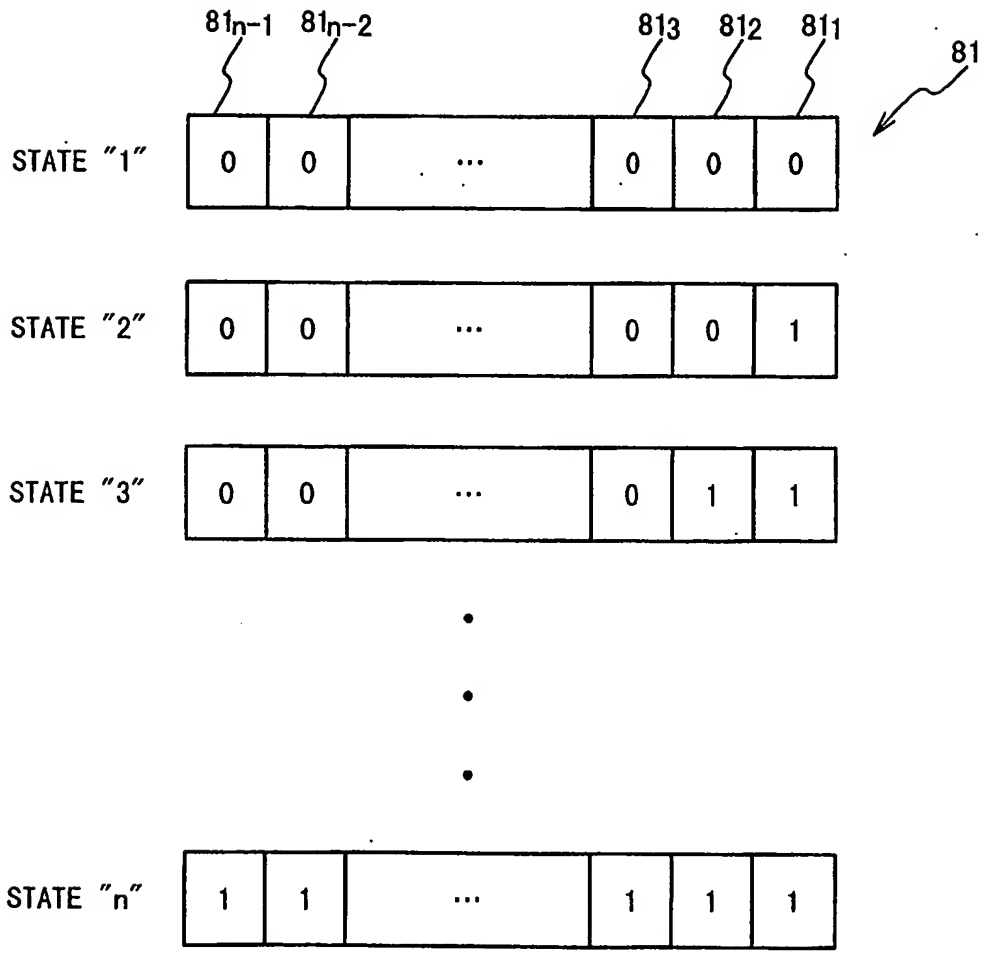

1
ROM
11
12
MAIN PROGRAM
ADDRESS CONVERSION PROGRAM
IN-ROM DATA
13
6
4
8
NON-VOLATILE MEMORY
POINTER FLAG REGION
POINTER FLAG
81
7
DATA REGION
71₁
MEMORY REGION
71₂
MEMORY REGION
71n
MEMORY REGION
RAM
31
IN-RAM DATA
3
CPU
2
POWER SUPPLY UNIT
5

Fig. 4


81n-1
81n-2
813
812
811
81
STATE "1"
0 0 ··· 0 0 0
STATE "2"
0 0 ··· 0 0 1
STATE "3"
0 0 ··· 0 1 1
STATE "n"
1 1 ··· 1 1 1

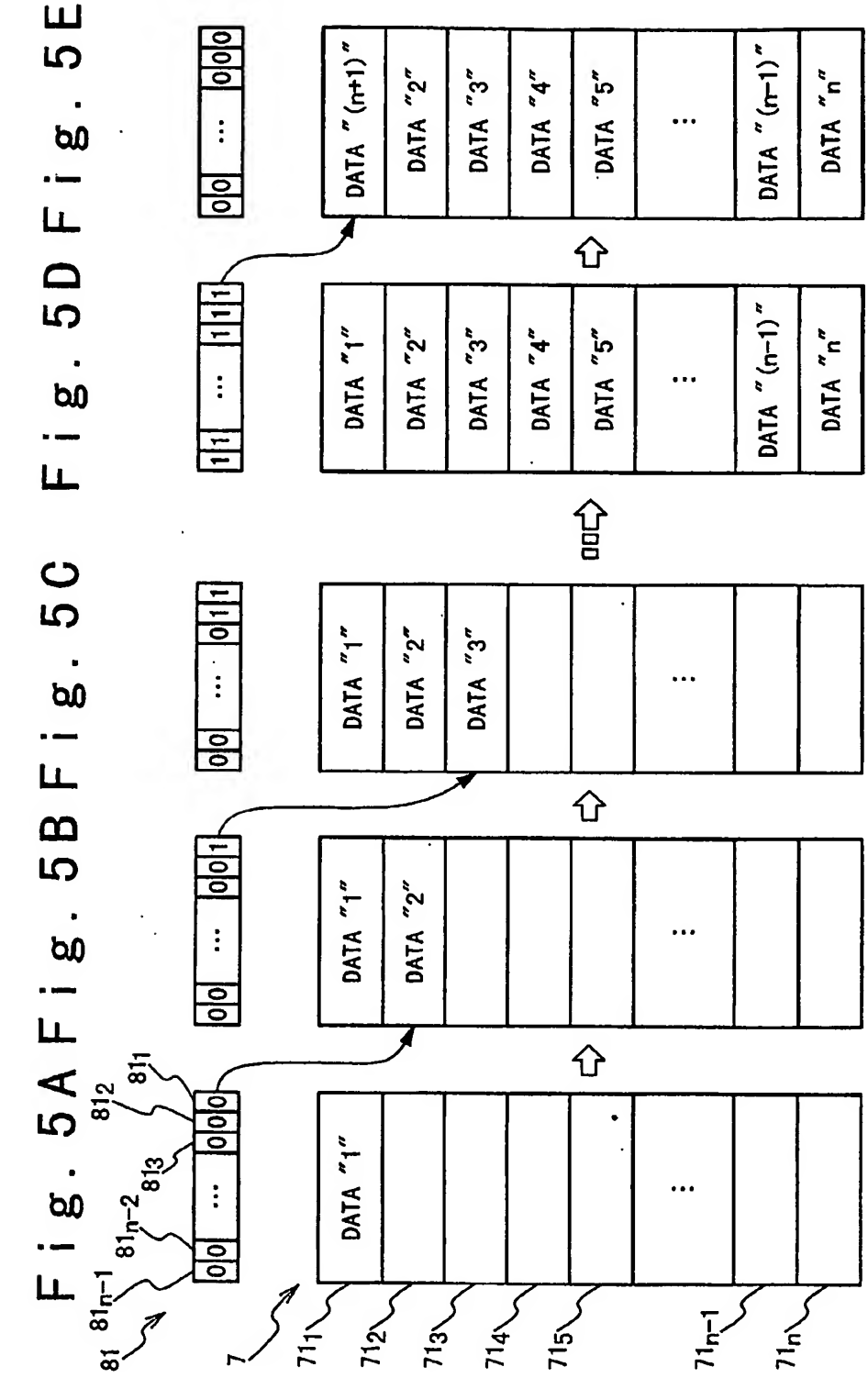


Fig. 5A Fig. 5B Fig. 5C Fig. 5D Fig. 5E
DATA "1"
DATA "2"
DATA "3"
DATA "4"
DATA "5"
DATA "(n-1)"
DATA "n"
DATA "(n+1)"
7
81
81n-1
81n-2
813
812
811
711
712
713
714
715
71n-1
71n

Fig. 6

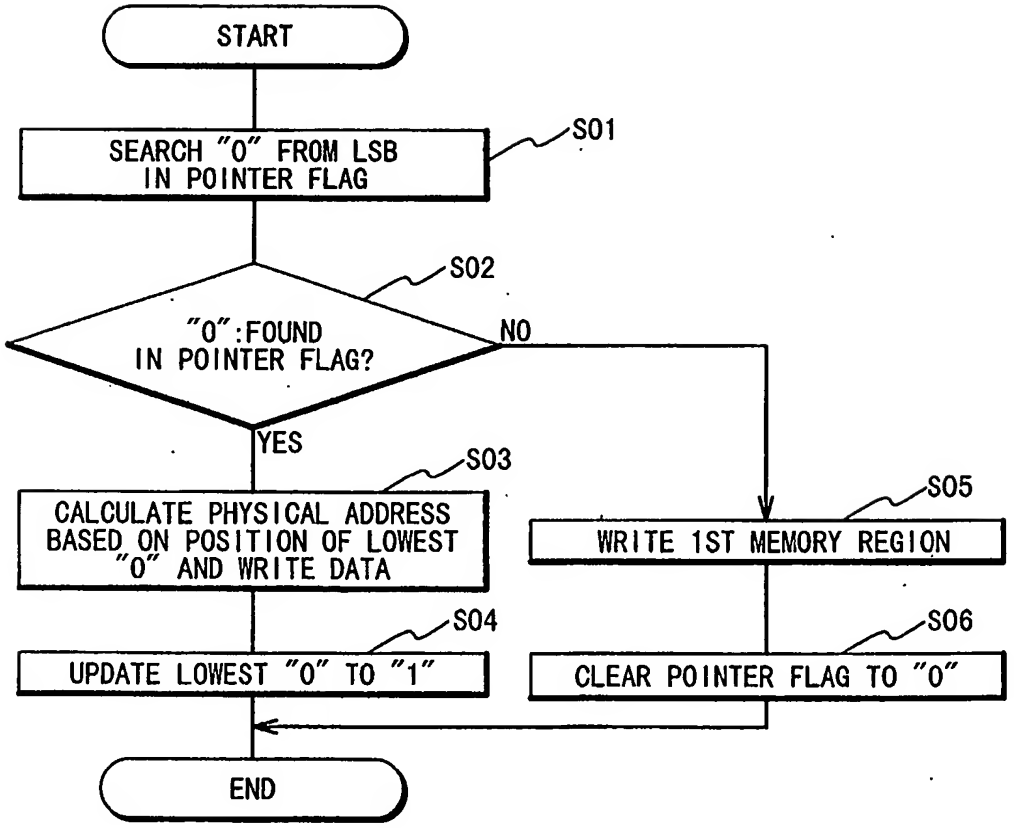

START
SEARCH "0" FROM LSB IN POINTER FLAG
S01
S02
"0":FOUND IN POINTER FLAG?
NO
YES
S03
CALCULATE PHYSICAL ADDRESS BASED ON POSITION OF LOWEST "0" AND WRITE DATA
S04
UPDATE LOWEST "0" TO "1"
S05
WRITE 1ST MEMORY REGION
S06
CLEAR POINTER FLAG TO "0"
END

Fig. 7A Fig. 7B Fig. 7C Fig. 7D Fig. 7E

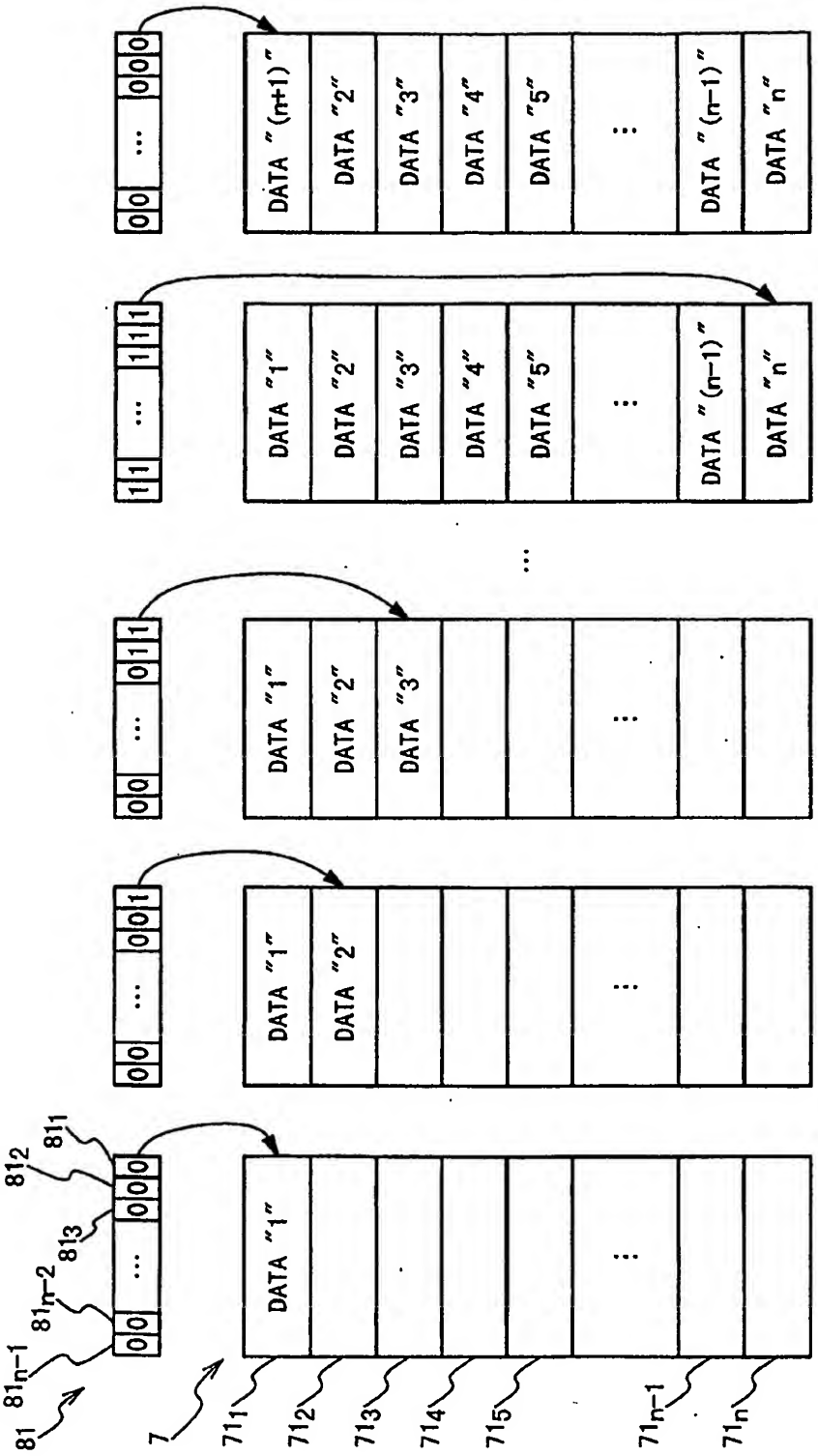


Fig. 8

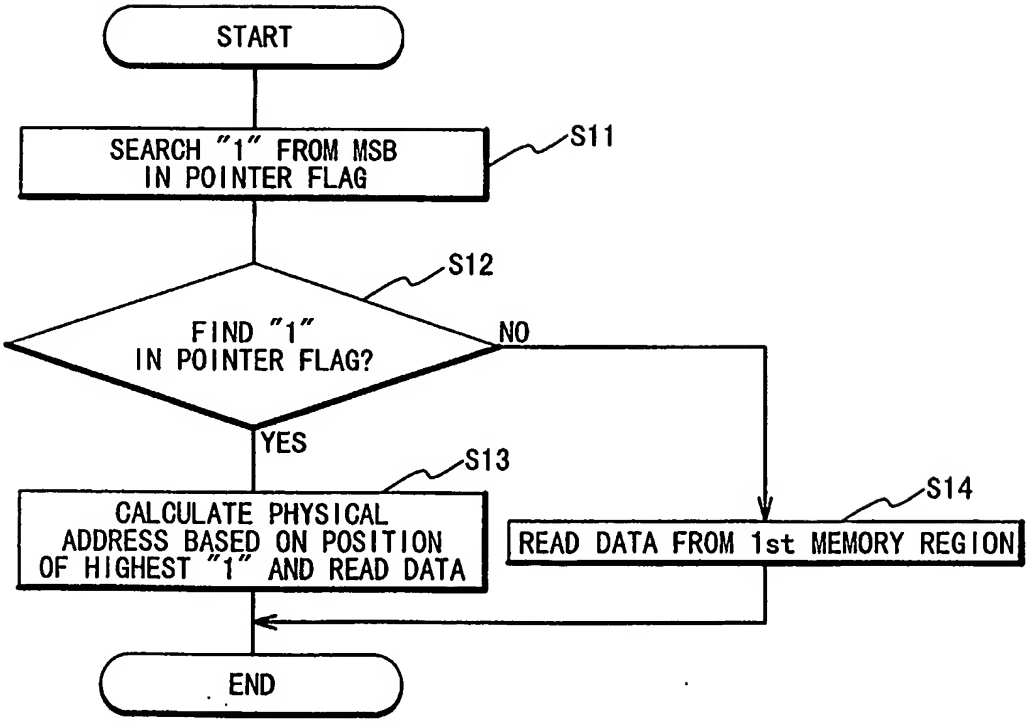

START
SEARCH "1" FROM MSB IN POINTER FLAG
S11
S12
FIND "1" IN POINTER FLAG?
NO
YES
S13
CALCULATE PHYSICAL ADDRESS BASED ON POSITION OF HIGHEST "1" AND READ DATA
S14
READ DATA FROM 1st MEMORY REGION
END

Fig. 9A

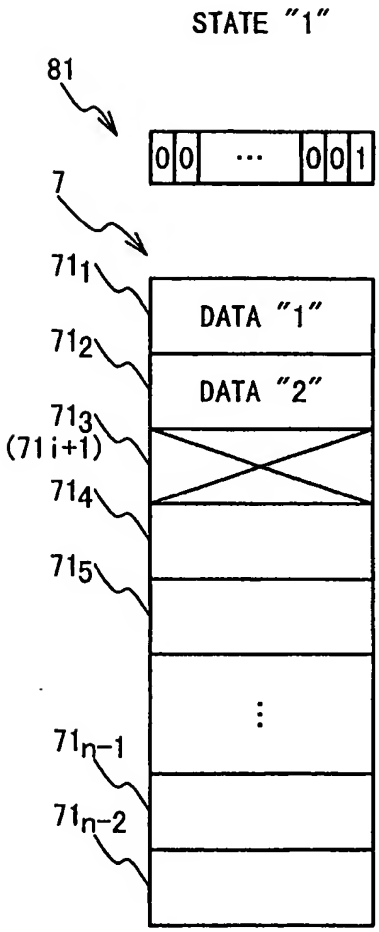

STATE "1"
81
0 0 ... 0 0 1
7
71 1
71 2
71 3
(71 i+1)
71 4
71 5
71 n-1
71 n-2
DATA "1"
DATA "2"

Fig. 9B

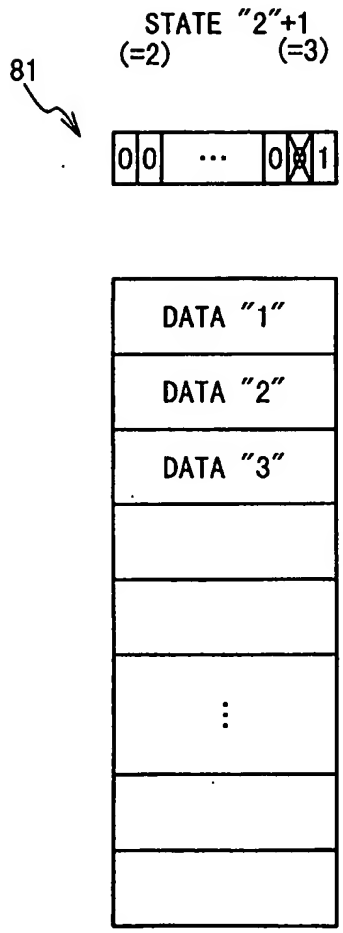

STATE "2"+1
(=2)
(=3)
81
0 0 ... 0 0 1
DATA "1"
DATA "2"
DATA "3"

Fig. 10

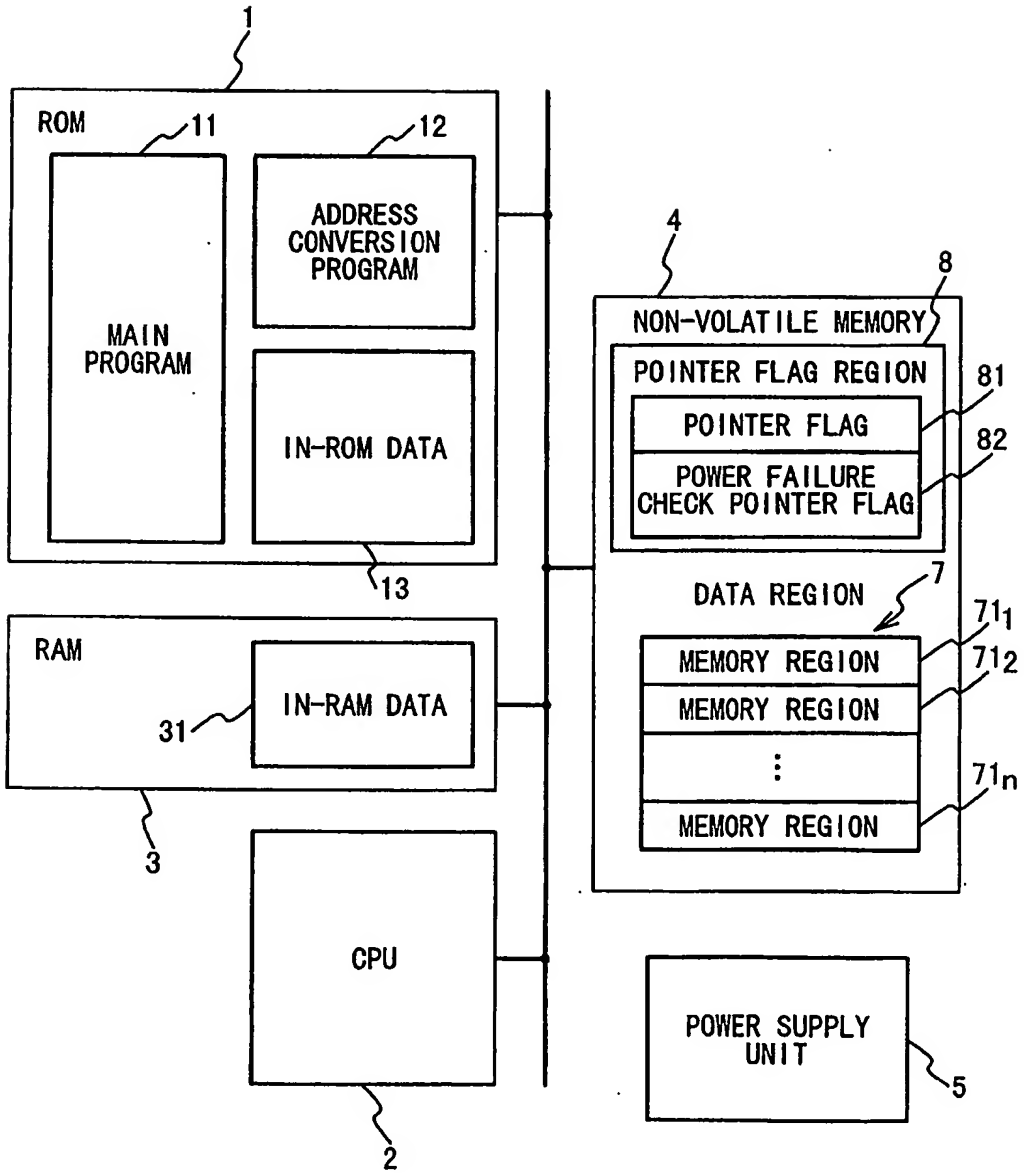
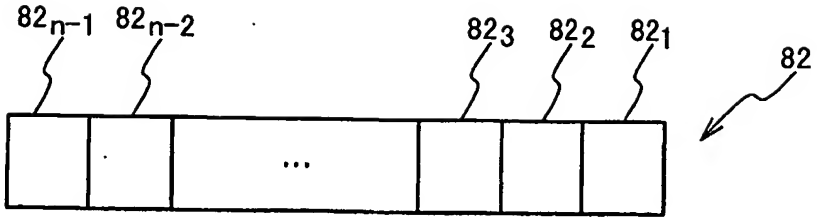

1
ROM
11
12
MAIN PROGRAM
ADDRESS CONVERSION PROGRAM
IN-ROM DATA
13
RAM
31
IN-RAM DATA
3
CPU
2
4
8
NON-VOLATILE MEMORY
POINTER FLAG REGION
81
POINTER FLAG
82
POWER FAILURE CHECK POINTER FLAG
DATA REGION
7
71 1
MEMORY REGION
712
MEMORY REGION
71n
MEMORY REGION
POWER SUPPLY UNIT
5

Fig. 11



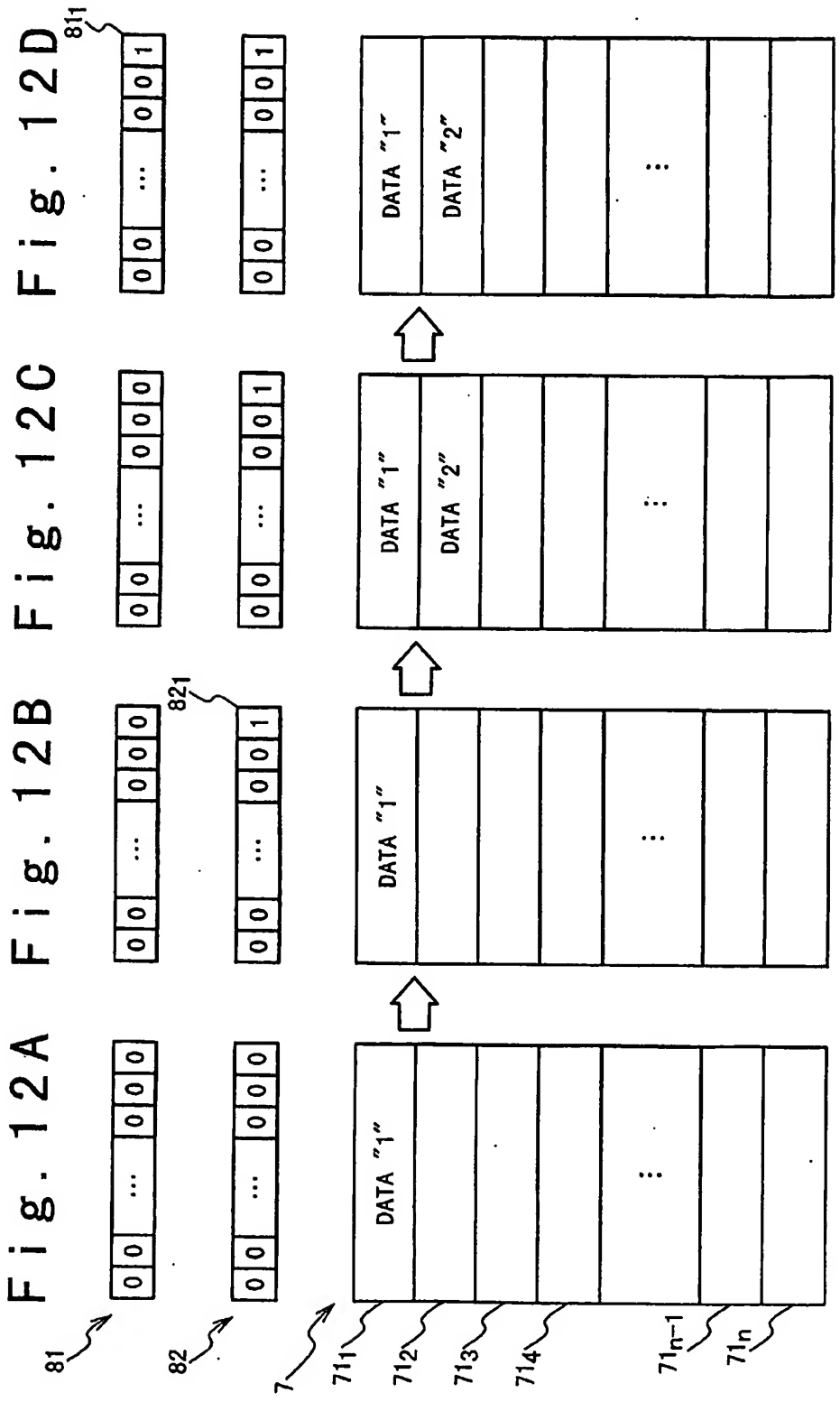


Fig. 12A
Fig. 12B
Fig. 12C
Fig. 12D
81
82
7
811
821
DATA "1"
DATA "2"
711
712
713
714
71n-1
71n

Fig. 13A Fig. 13B Fig. 13C Fig. 13D

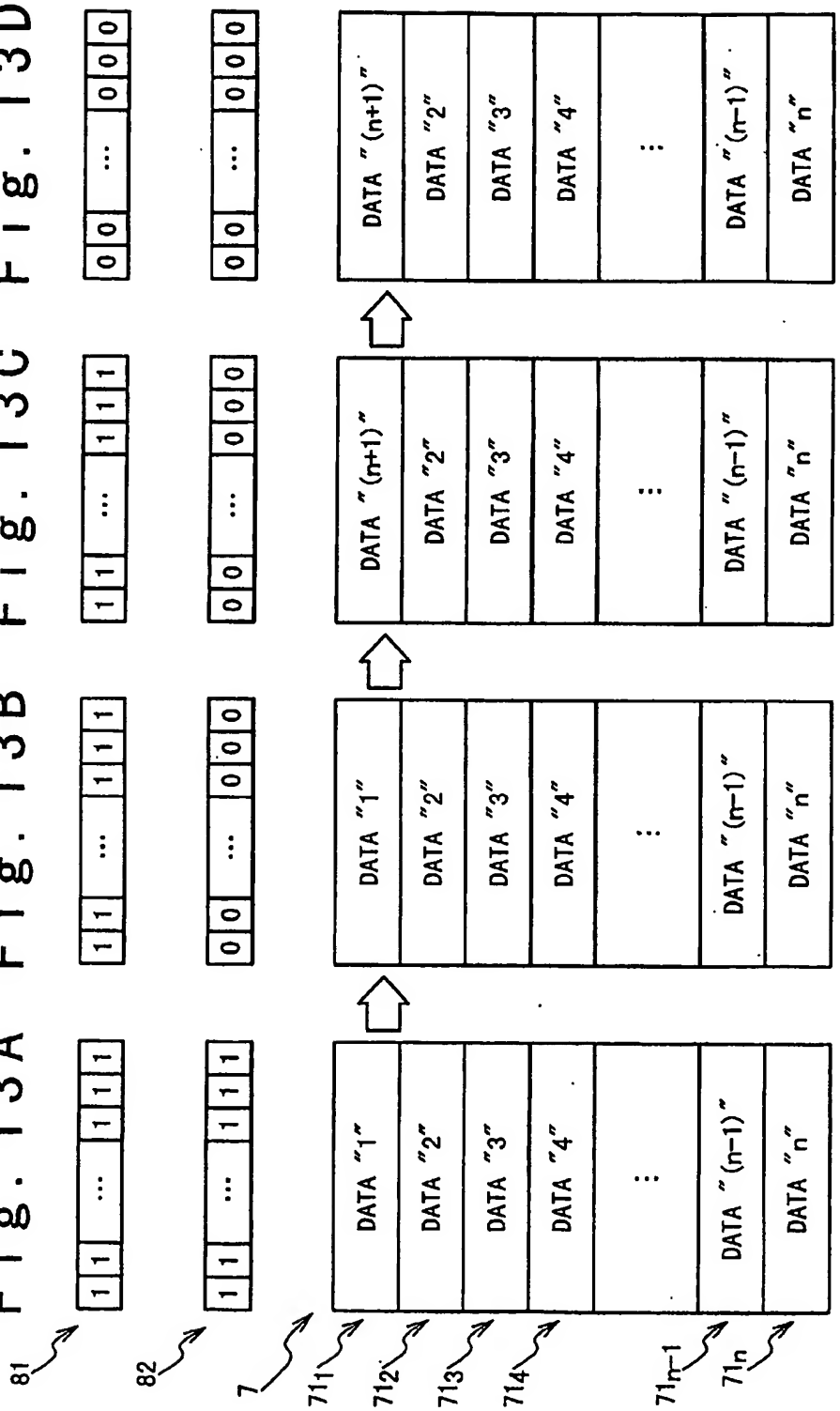


Fig. 14

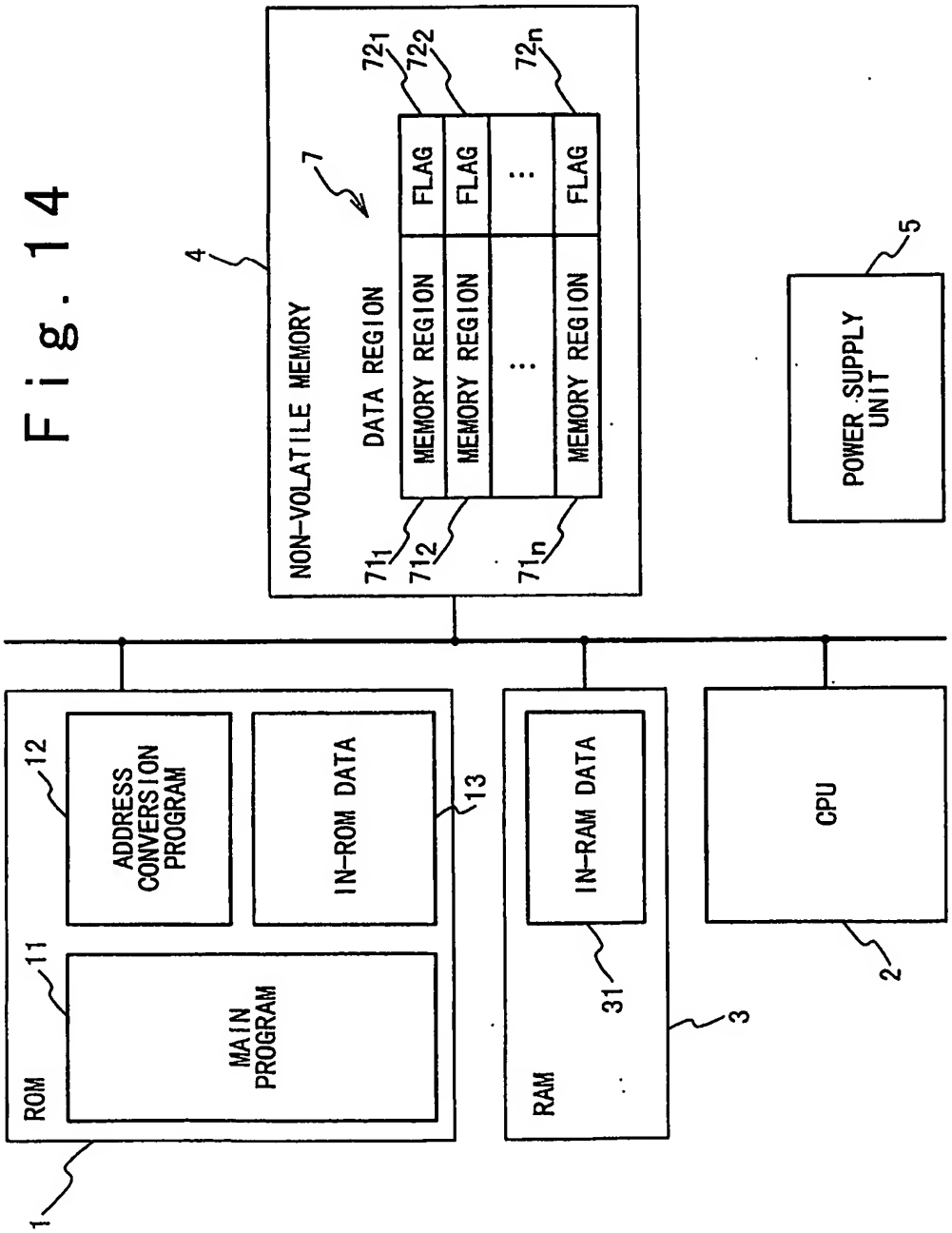
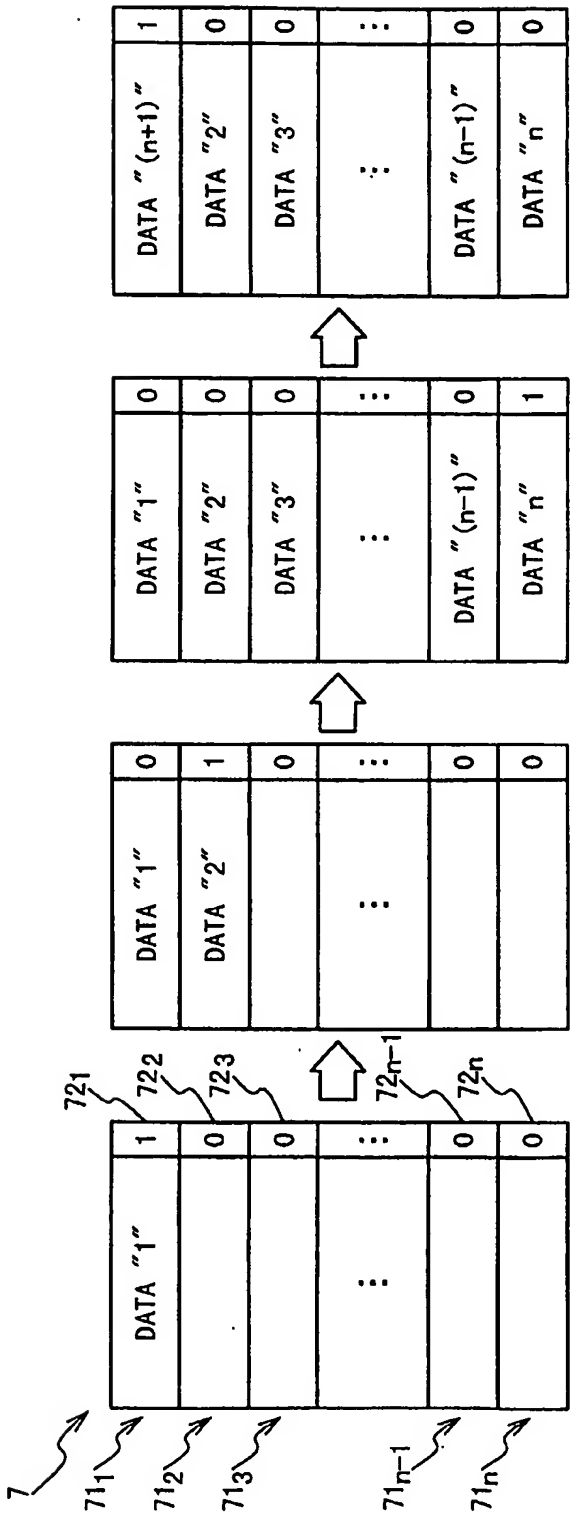

NON-VOLATILE MEMORY
4
7
DATA REGION
721
722
72n
FLAG
FLAG
FLAG
MEMORY REGION
MEMORY REGION
MEMORY REGION
711
712
71n
POWER SUPPLY UNIT
5
ROM
11
ADDRESS CONVERSION PROGRAM
12
IN-ROM DATA
13
MAIN PROGRAM
1
RAM
IN-RAM DATA
31
3
CPU
2

Fig. 15A Fig. 15B Fig. 15C Fig. 15D



European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 00 5757

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	FR 2 665 791 A (MAZINGUE DIDIER;REMERY PATRICK) 14 February 1992 (1992-02-14)	1-9, 12-19, 21,22	G11C16/34 G11C16/10
Y		10,20	
A		11	
	* page 3, line 22 - page 6, line 33 * * tables I-IV *		
X	US 4 780 855 A (IIDA NORIHIKO ET AL) 25 October 1988 (1988-10-25)	1-4,8,9, 12, 14-16, 19,21,22	
Y		10,20	
A		5-7,11, 13,17,18	
	* column 2, line 41 - column 3, line 53 * * figure 1 *		
Y	FR 2 730 833 A (GEMPLUS CARD INT) 23 August 1996 (1996-08-23) * page 11, line 6 - page 14, line 4 * * figures 2,3 *	10,20	
X	PATENT ABSTRACTS OF JAPAN vol. 016, no. 182 (P-1346), 30 April 1992 (1992-04-30) -& JP 04 021999 A (MITSUBISHI ELECTRIC CORP), 24 January 1992 (1992-01-24)	1-4,8,9, 12, 14-16, 19,21,22	
Y	* abstract *	10,20	
X	FR 2 712 412 A (CITROEN SA;PEUGEOT) 19 May 1995 (1995-05-19) * page 3, line 22 - page 5, line 18 * * figures 2,3 *	1-4, 14-16	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)
G11C

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	24 June 2002	Gaertner, W

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 02 00 5757

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

24-06-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
FR 2665791 A	14-02-1992	FR 2665791 A1	14-02-1992
US 4780855 A	25-10-1988	JP 1728759 C	29-01-1993
		JP 4011957 B	03-03-1992
		JP 61008798 A	16-01-1986
FR 2730833 A	23-08-1996	FR 2730833 A1	23-08-1996
		AU 691406 B2	14-05-1998
		AU 4835296 A	04-09-1996
		DE 69609256 D1	17-08-2000
		DE 69609256 T2	18-10-2001
		EP 0756746 A1	05-02-1997
		ES 2150104 T3	16-11-2000
		WO 9625743 A1	22-08-1996
		JP 2846739 B2	13-01-1999
		JP 9506460 T	24-06-1997
		ZA 9601123 A	23-08-1996
JP 04021999 A	24-01-1992	NONE	
FR 2712412 A	19-05-1995	FR 2712412 A1	19-05-1995

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82